

AD-A229 121

DTIC FILE COPY

Technical Document 1830
June 1990

Software Performance Engineering for Realtime Systems

L&S Computer Technology, Inc.

DTIC
SELECTED
NOV 03 1990
8 D

Approved for public release; distribution is unlimited.

The views and conclusions contained in this report are those of the contractors and should not be interpreted as representing the official policies, either expressed or implied, of the Naval Ocean Systems Center or the U.S. Government.

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

Contract N00039-86-C-0247 was carried out by L&S Computer Technology, Inc., P.O. Box 9802, Austin, TX 78766, for the Office of Naval Technology, Arlington, VA 22217, under the technical coordination of T. Sterrett, Computer Systems Software and Technology Branch, Code 411, Naval Ocean Systems Center, San Diego, CA 92152-5000.

Released by
R. A. Wasilausky, Head
Computer Systems Software
and Technology Branch

Under authority of
A. G. Justice, Head
Information Processing and
Displaying Division

NOTE

Permission to print copyrighted material for government purposes has been granted by the author, Dr. C. U. Smith.

FS

AGENDA

in the following times:

Day One

8:30-9:30

INTRODUCTION

What is SPE, What is POD,

Role of modeling in the lifecycle vs. fix it later,

Overview of class

9:30-10:00

SOFTWARE PERFORMANCE MODELING INTRO

Data Requirements: performance goals, workload, environment,
software structure, resource requirements,

10:00-10:20

Break

10:20-11:30

SOFTWARE PERFORMANCE MODEL REPRESENTATION

Execution graph software models

Exercise

11:30-1:00

Lunch

1:00-2:10

SOFTWARE PERFORMANCE MODEL EVALUATION

Exercise

2:10-2:30

Break

2:30-3:00

INTRODUCTION TO POD

3:00-4:30

LAB EXERCISE I. POD Tool introduction

Problem 1: Static analysis

Problem 2: Revise software

Day Two

8:30-8:45 Review

8:45-9:15 ROLE OF STATIC ANALYSIS IN SOFTWARE LIFECYCLE,

9:15-10:00 SYSTEM MODELING I
Introduction to operational analysis

10:00-10:20 Break

10:20-11:30 SYSTEM MODELING II
Exercise
Introduction to POD capabilities

11:30-1:00 Lunch

1:00-4:30 SYSTEM MODELING WORKSHOP
Tool Introduction - PC
Problem 1: Create & Solve Simple Model
Problem 2: More complex model
Problem 3: Modeling Design Alternatives

Day Three

8:30-8:45 Review

8:45-10:10 SPE IN THE LIFECYCLE: (KR)
Overview of activities
Early lifecycle example
(FAA examples)

10:10-10:30 Break

10:30-11:30 SPE IN THE LIFECYCLE (cont.):
Mid lifecycle example:
(still looking for a good example - will try for a variation of OTH)

Late lifecycle examples:
(add algorithm F/A18, cycle time investigation, manipulate priorities)

11:30-1:00 Lunch

- 1:00-2:00 OVERVIEW OF ADVANCED MODELS
 General methods
 (Approximations, shadow servers, iterative solutions)
- 2:00-2:15 Break
- 2:15-3:15 Concurrent processes
 Serialization delays
 Memory modeling
- 3:15-3:30 Break
- 3:30-4:30 Advanced models cont.

Day Four (3-1/2)

- 8:00-9:30 GENERAL PRINCIPLES FOR LIFECYCLE OPTIMIZATION
- 9:30-9:50 Break
- 9:50-11:00 INTEGRATION
 Relationship to Software development methods and tools
 PE Implementation Considerations
 (discuss viability of SPE for attendee's environments)
 Summary
 Question/Answer session



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

Introduction & Class Overview

LS

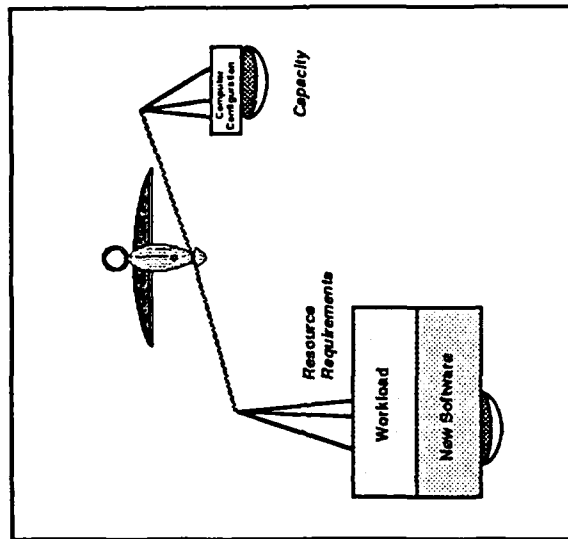
Introduction and Class Overview

PART 1

What is
Software
Performance
Engineering ?

L S 1.1.1

PERFORMANCE BALANCE



L S 1.1.2

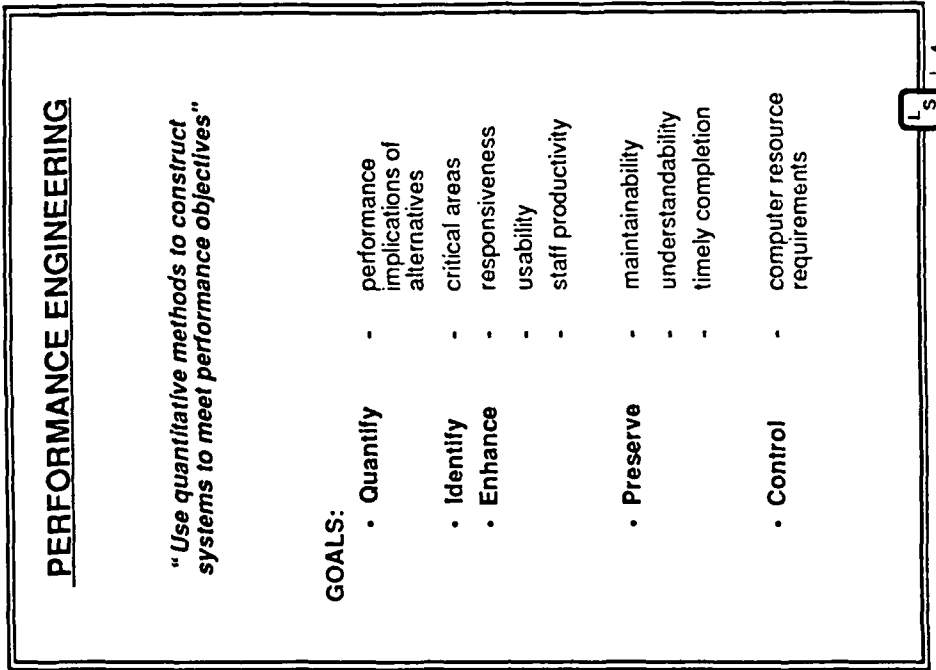
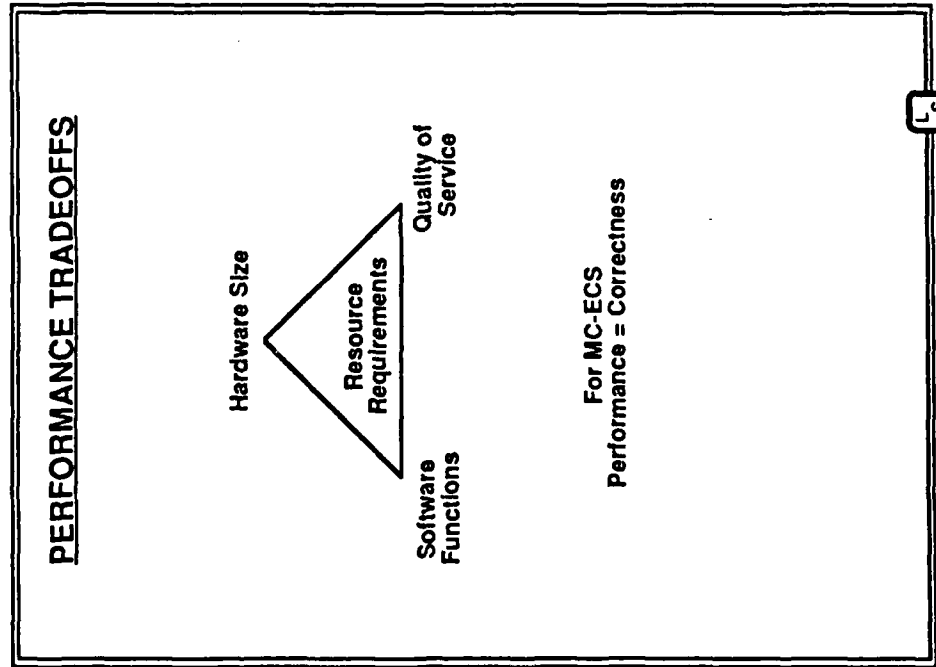
I - 1

Notes

LS

Introduction and Class Overview

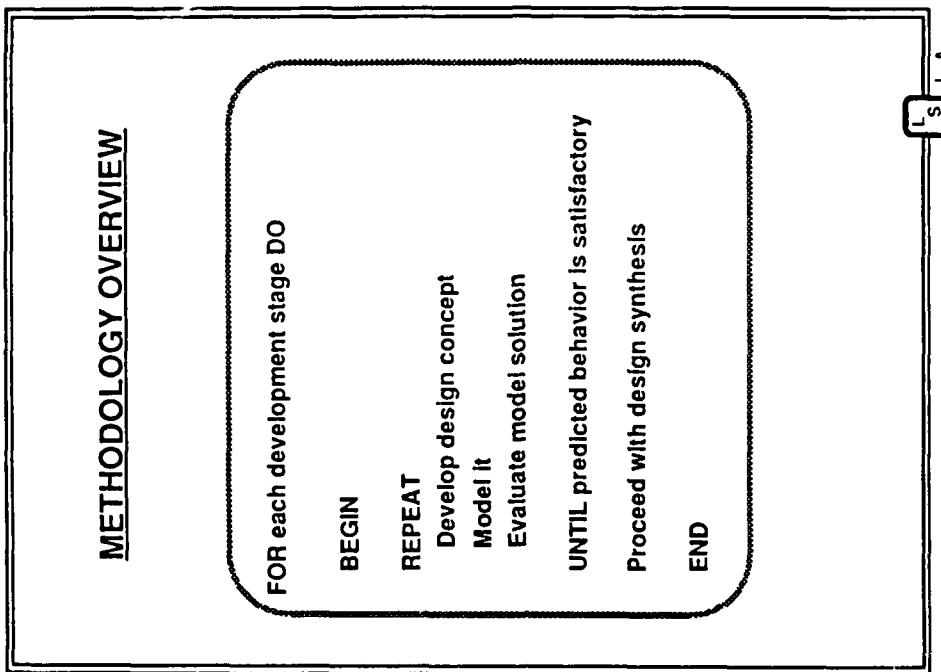
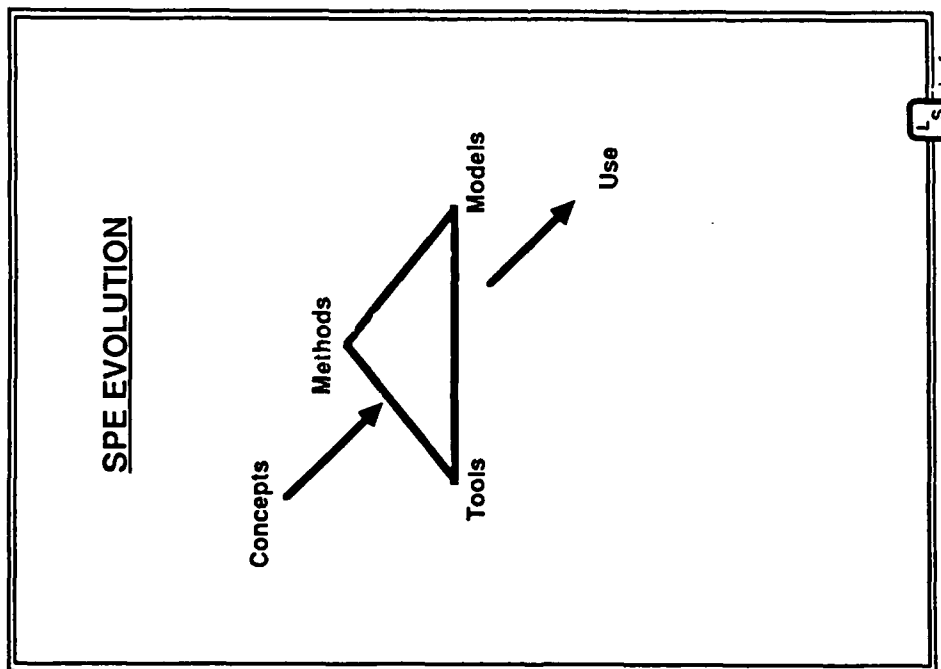
∞



Notes

L's

Introduction and Class Overview

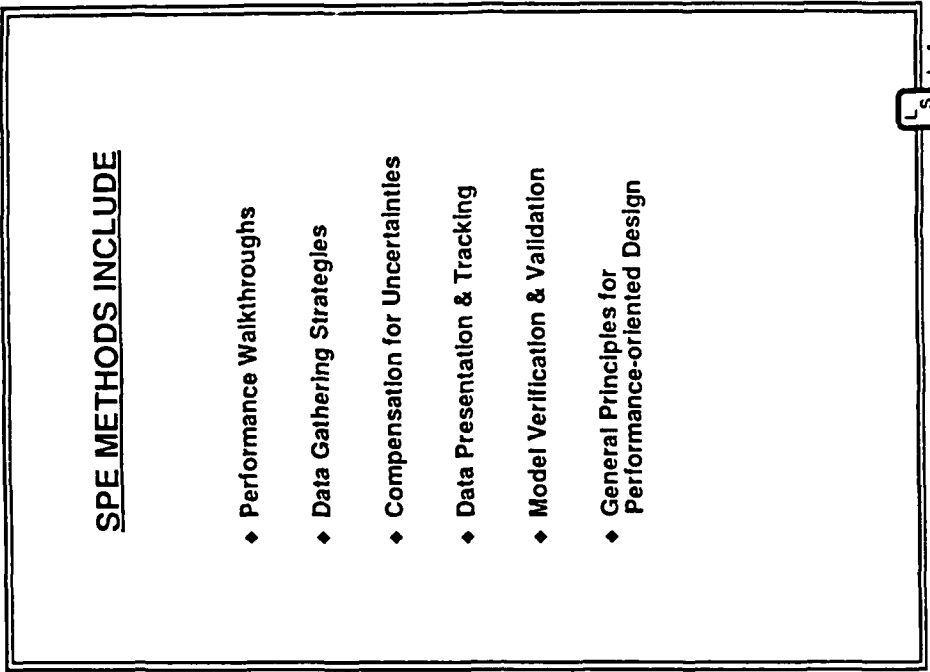
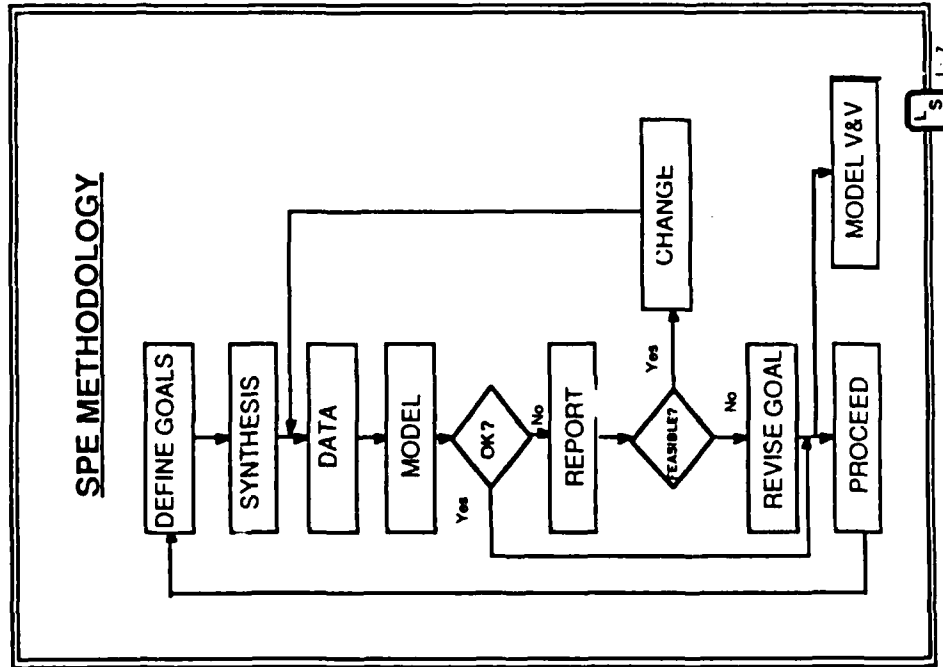


I - 3

Notes

L S

Introduction and Class Overview



I - 4

Notes

L's

Introduction and Class Overview

MODELING REQUIREMENTS

- ♦ Low overhead
- ♦ Accommodate:
Incomplete definitions
Imprecise performance specs
Changes during development

GOAL:

Initially

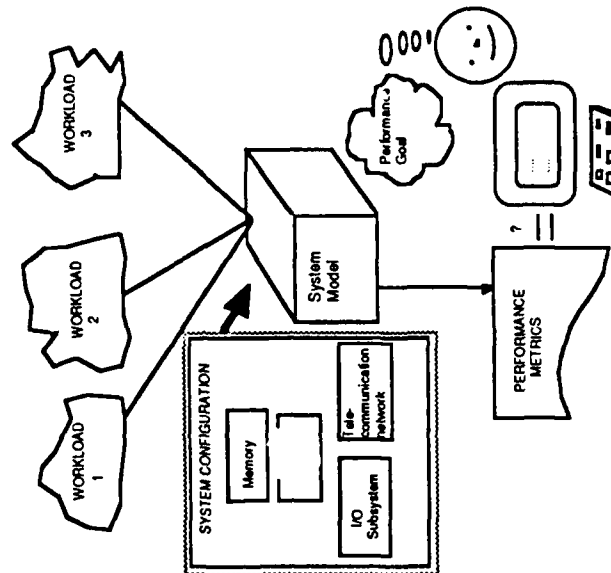
- Distinguish between "good" and "bad"

Later

- Increase precision of predictions
- Provide implementation decision support
- Identify problem areas
- Provide guidance for testing

L S 1-4

CONVENTIONAL PERFORMANCE MODELING



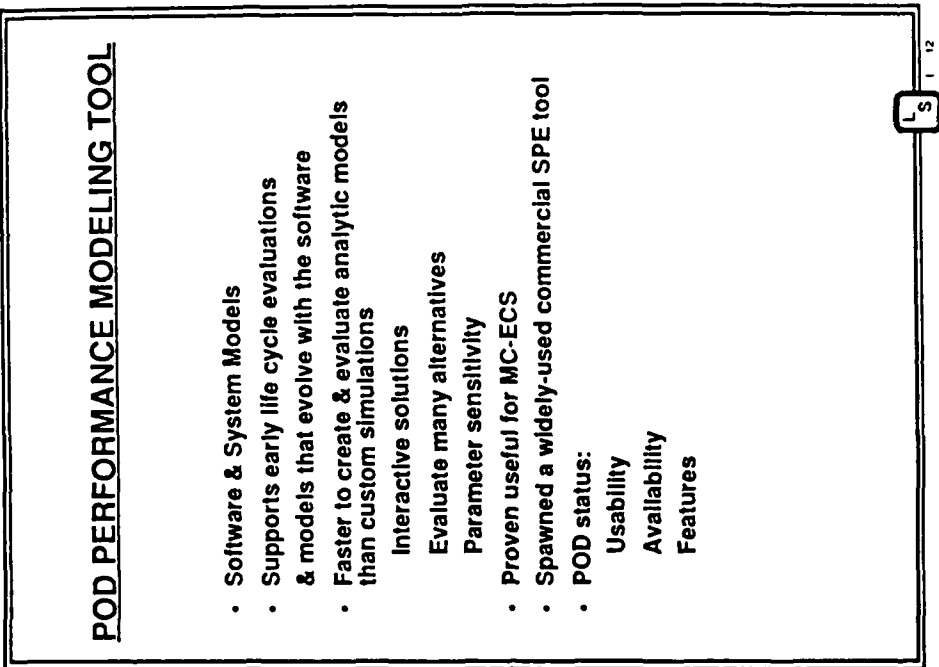
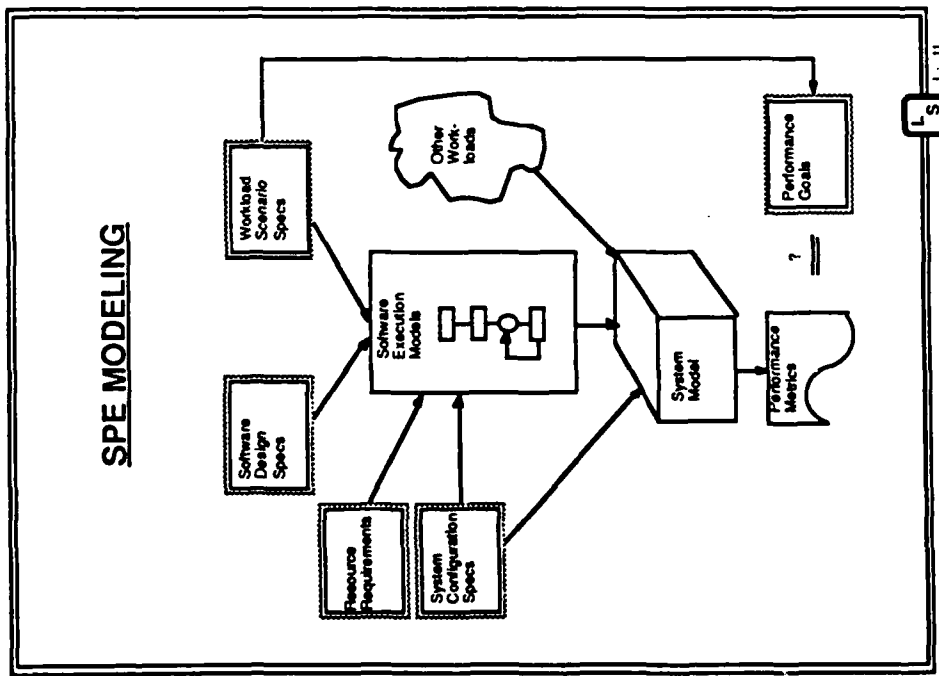
L S 1-5

I - 5

Notes

15

Introduction and Class Overview



I - 6

Notes

LS

Introduction and Class Overview

SPE STATUS

Modeling Expertise Easily Developed

Support Tools Available

- Performance Modeling
- Computer System Measurement

SPE Usage

- Many successful experience papers emerging
- SPE Bibliography

OVERVIEW

I. Background

What is Software Performance Engineering?

Who is here?

SPE experience?

Expectations?

II. Software Performance Models

What data is needed?

What is the first modeling step?

How do you handle data dependency?

III. Workshop: Software Performance Models

POD Introduction

Create model

Evaluate alternatives

Notes

19

Introduction and Class Overview

OVERVIEW - cont.

IV. Role of Static Analysis In Software Life Cycle

When is it useful?

What does it tell you?

Recommendations

V. System Execution Models

How do you analyze the effects of computer resource contention?

How are more complex questions about the software answered?

Modeling fundamentals

VI. System Model Workshop

POD Performance calculator

Experience creating and solving models

Modeling design alternatives

L S

1 11

OVERVIEW - cont.

VII. SPE in the Life Cycle

SPE Methods

Overview of Activities

VIII. Life Cycle Examples

Early life cycle

Mid life cycle

Late life cycle

IX. Overview of Advanced Models

How can you use analytic models to study intricate processing details?

General methods - approximations, iterative solutions, shadow servers.

Specific analyses - concurrent processes, federated systems, serialization delays.

L S

1 11

I - 8

Notes

19

Introduction and Class Overview

OVERVIEW cont.

X. General Principles for Life Cycle Optimization

What are some techniques for designing efficient systems?

How do you teach others about performance-oriented design?

XI. Integration of Software Performance Engineering

What are some other relevant software engineering breakthroughs?

Are they compatible with SPE?

Review - SPE/POD Technology Transfer Project & Results

XII. Summary

Is SPE viable in your environment?

What new features do you need?

Question / Answer session

OVERVIEW - cont.

Many examples and exercises are included. Learn by doing!

Class participation is encouraged!

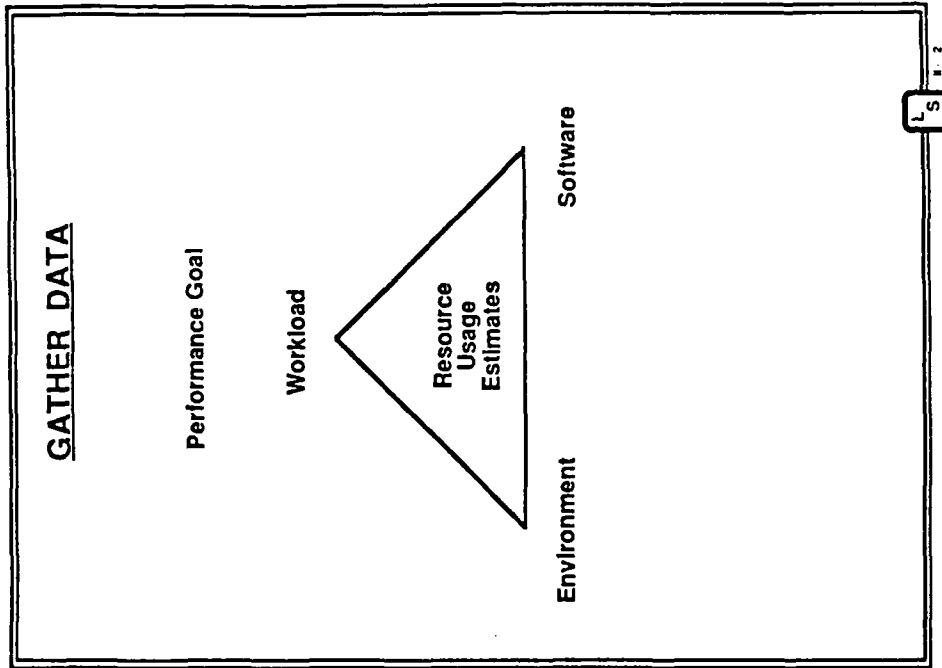
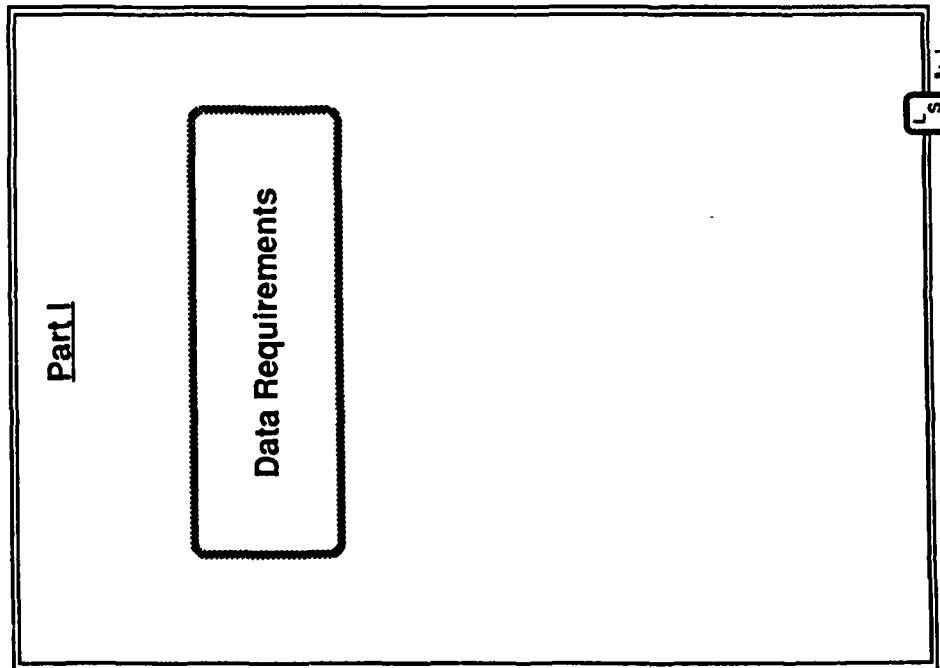
Suggestions are welcome!

I - 9

Software Performance Models

2

Software Performance Models



II - 1

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

L^s

Software Performance Models

WORKLOAD DATA

- ♦ Design MC-ECS to meet worst-case performance goals
- ♦ Typical-case performance: Pareto principle ('80-20 rule')
Less than 20% of the functions of the system are executed more than 80% of the time
- ♦ Scenarios define cases of Interest
 - Performance goals
 - Update rates
 - End to end tasks
 - Timing errors?
 - Request arrival rates
 - Number of concurrent users

L S 1.3

EXAMPLE - F/A - 18 OFF

System Functions:

Navigation Point to point Auto Pilot Data Link	Man-Machine Interface Navigation Support Tactical
Weapon Delivery Air to air Sparrow Gun Sidewinder Air to ground Nuclear Conventional	Target Detection & Designation Air to air Radar Visual Data Link Air to ground Radar LST Data link FLIR Visual TV

L S 1.4

11 - 2

Notes

L^S

Software Performance Models

OFP WORST CASE SCENARIO

Air to air mode
Single target track
Track while scan
Sparrow mode selected
Data link targets up
.
.
.

Scenario Questions

How to identify scenario modes and longest paths?

How to create representative models that are simpler than the software?

PERFORMANCE QUESTIONS

- Can the ___ processor handle the job?
Memory sufficient?
Computation time?
- If not, what functions must be off loaded (or run at a lower update rate)?
- Should a function use table lookup or direct computation?
- Does the function cause timeouts in other "unrelated" functions?

Others?

11 - 3

Notes

LS

Software Performance Models

EXAMPLE - Surveillance System

Monitor Initialize Track file mgt. Sensor data mgt. Validate & correlate data Display control Track summary Tracks to combat control Track for quality check Tracks to communicate General display	Tactical Data Link-II Data transfer Display link data
Operator Training Data Links Commands	Operational Data Link Remote platforms Communication sites
Operating System Realtime control Centralized services	Combat Control System Fire control etc.

WORKLOAD SOURCES

- ◆ Specifications
- ◆ Measurements
 analogous systems
 prototypes
- ◆ Benchmarks (simulators)
- ◆ Interviews
- ◆ Walkthroughs

Who are the 'users' ?

11 - 4

Notes

19

Software Performance Models

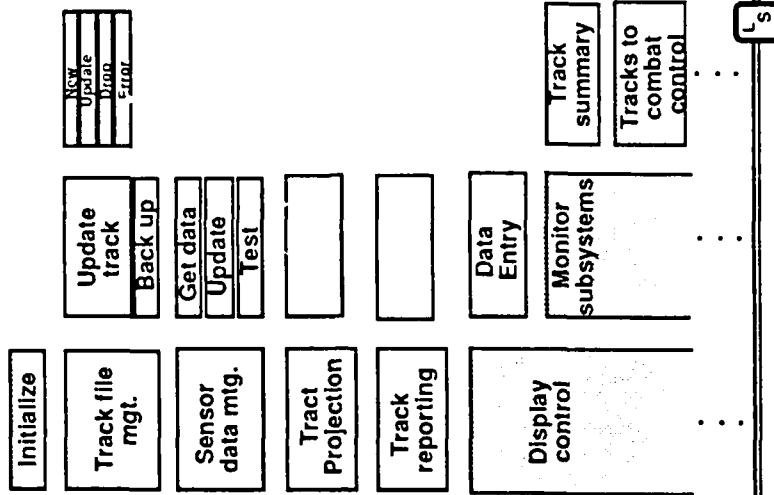
SOFTWARE SPECIFICATIONS

- ♦ Execution paths for scenarios
- ♦ Programs / modules / procedures / code fragments to be executed
probability of execution
number of repetitions
protocol
- ♦ Level of detail increases as development progresses

L S

SURVEILLANCE EXAMPLE cont.

Monitor Structure:



11 - 5

Notes



Software Performance Models

SOFTWARE EXECUTION MODELS

- ♦ Processing scenario:
 1. Initialize
 2. Display control
 Monitor subsystems
 Tracks to combat control
 .
 .
 .
- ♦ Performance goal:
 - seconds to initiate transfer
 - seconds to complete transfer
- ♦ Additional workload specifications:
 - 1 Operator console
 - 6 Requests per minute

L S # 11

ENVIRONMENT

- ♦ System characteristics:
 - Configuration
 - Device service rates
- ♦ Overhead for:
 - Lower ' level ' services
- ♦ Scheduling policies

*What if the project includes
hardware development?*

*What if the service routines
do not yet exist?*

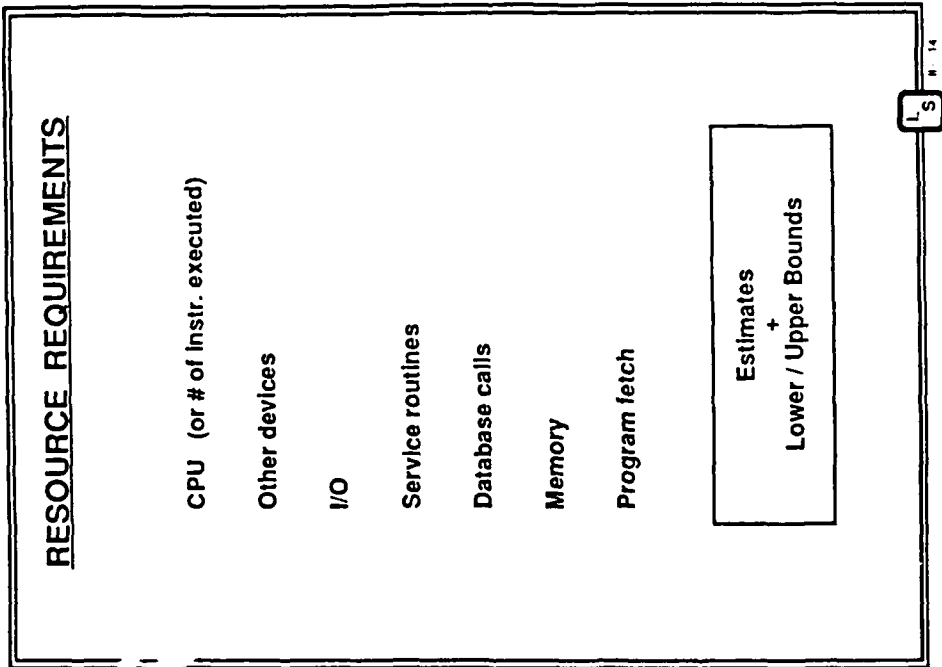
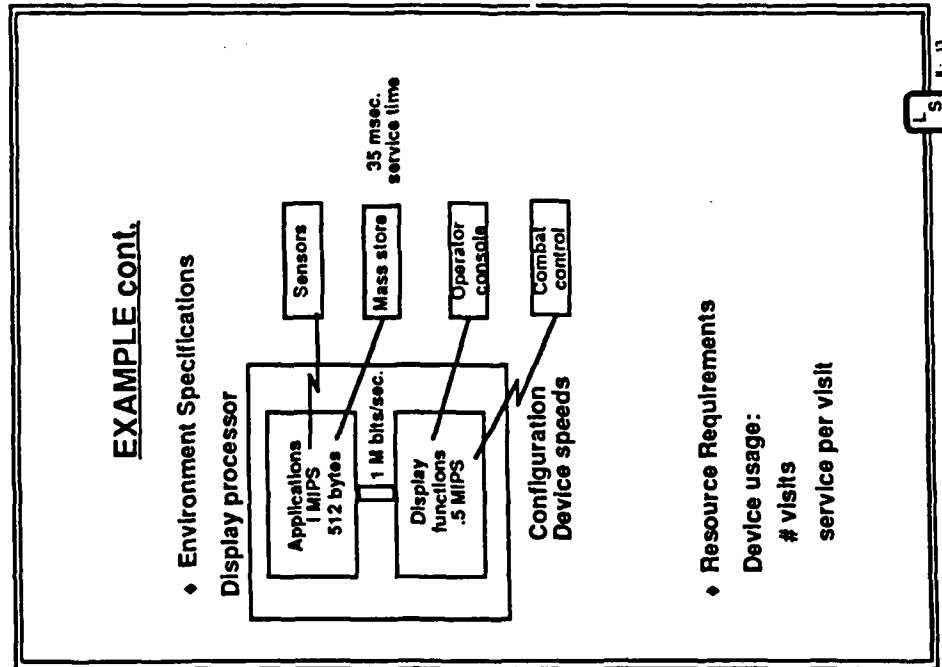
L S # 12

11 - 6

Notes

L'S

Software Performance Models



11 - 7

Notes

19

Software Performance Models

DATA PRECISION

- ♦ *Representative* workloads
- ♦ Use estimates & bounds to identify critical areas
- ♦ If imprecise:
 - Implement & measure
 - Construct benchmarks
 - Monitor development closely
 - Report precision of analysis

L S 11-18

Part II

Software Models: Representation

L S 11-18

11 - 8

Notes

L'S

Software Performance Models

SOFTWARE MODELS

- ♦ Derive software execution characteristics
- ♦ Initial check of performance goals
- ♦ Provide workload parameters for the system execution model

L S 17

EXECUTION GRAPH MODELS

- ♦ Visual representation of software
Model confirmation
Presentation of results
- ♦ Formal basis for analysis
Aids in comprehension
Specification for tool development
- ♦ Viable alternative to commercial tools

L S 18

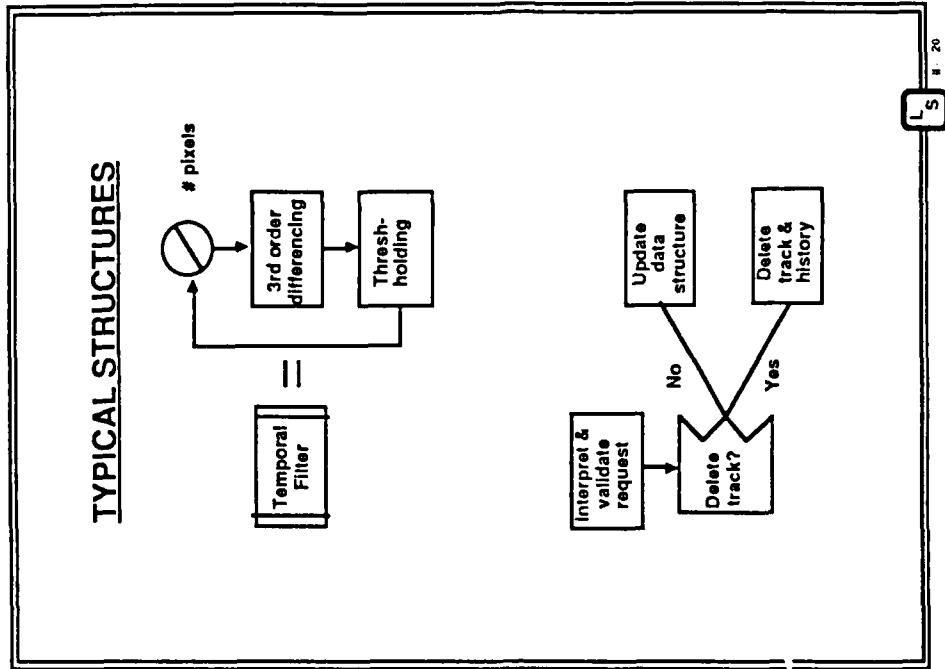
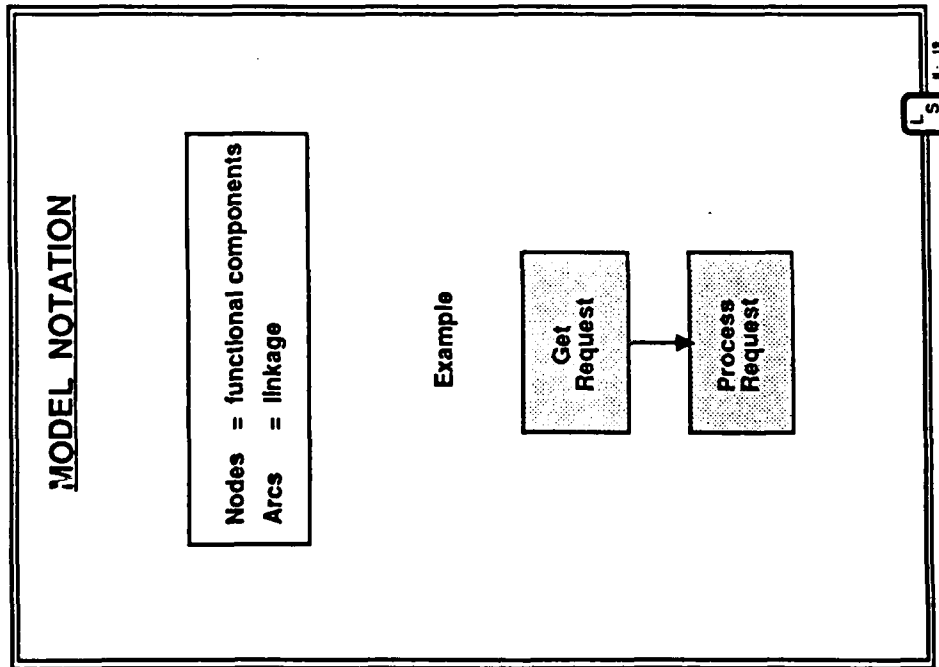
11 - 9

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

15

Software Performance Models

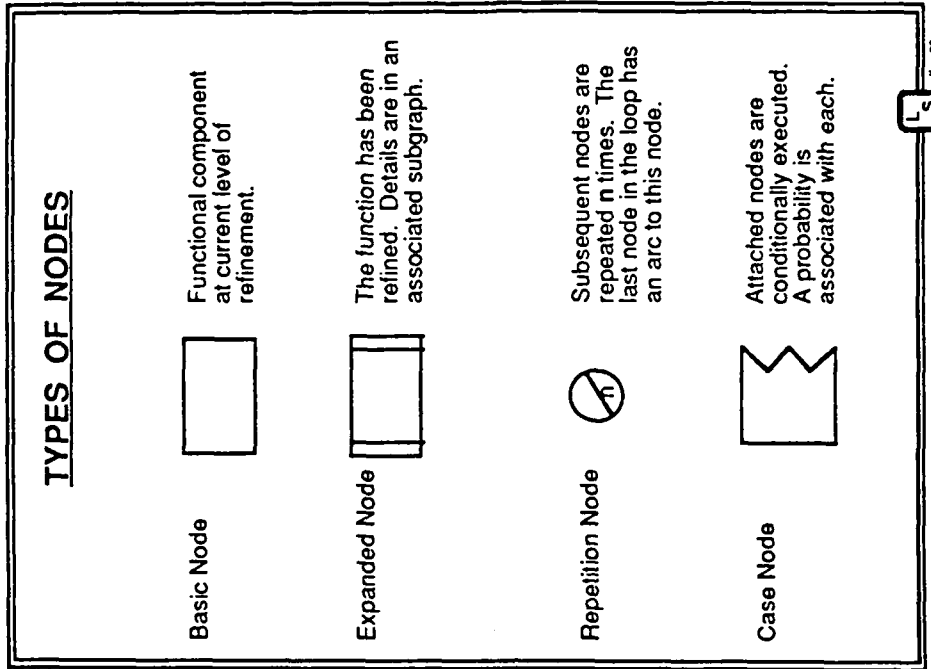
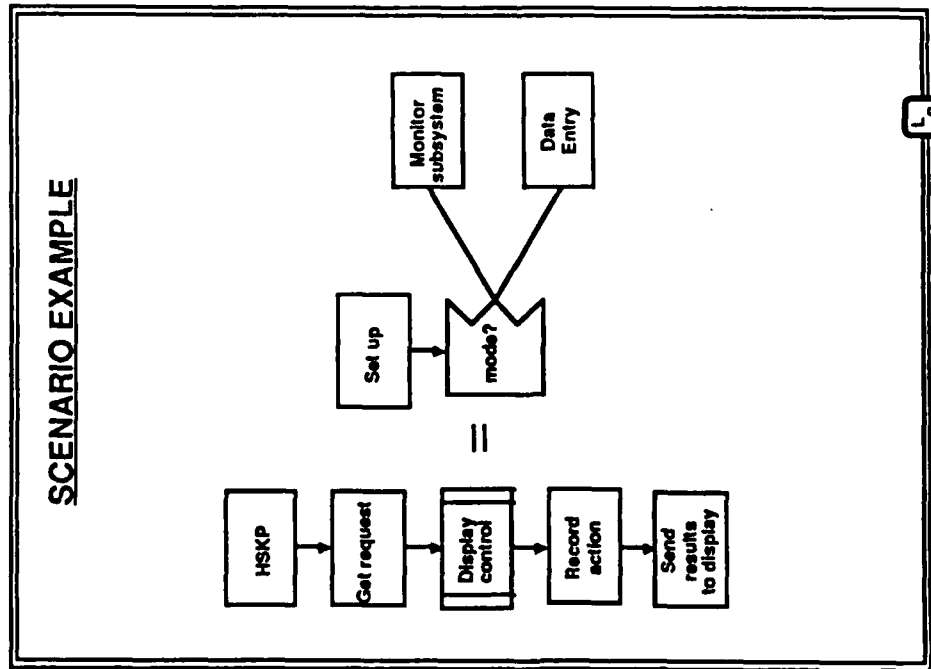


II - 10

Notes



Software Performance Models

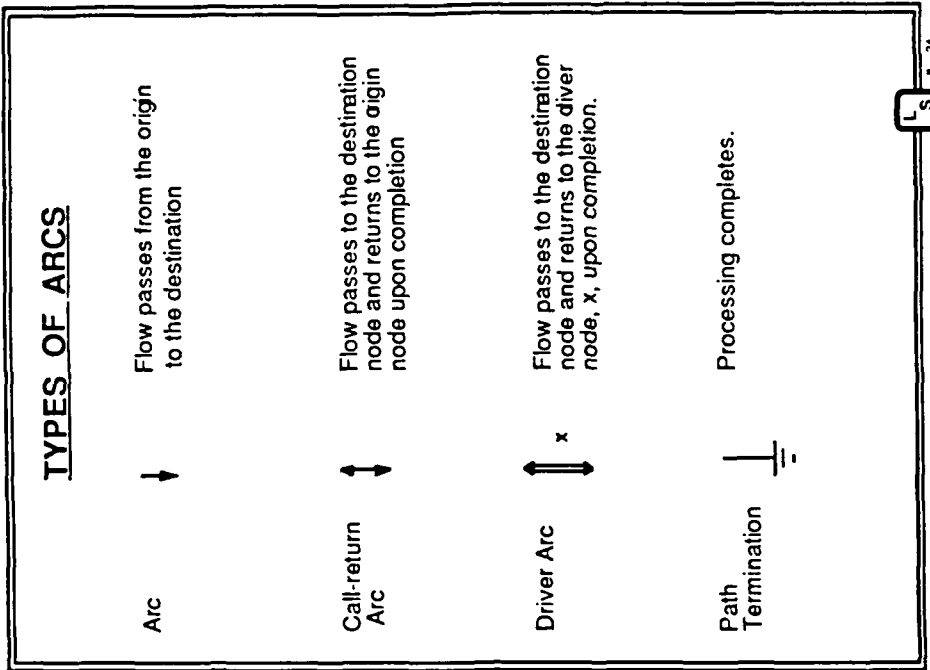
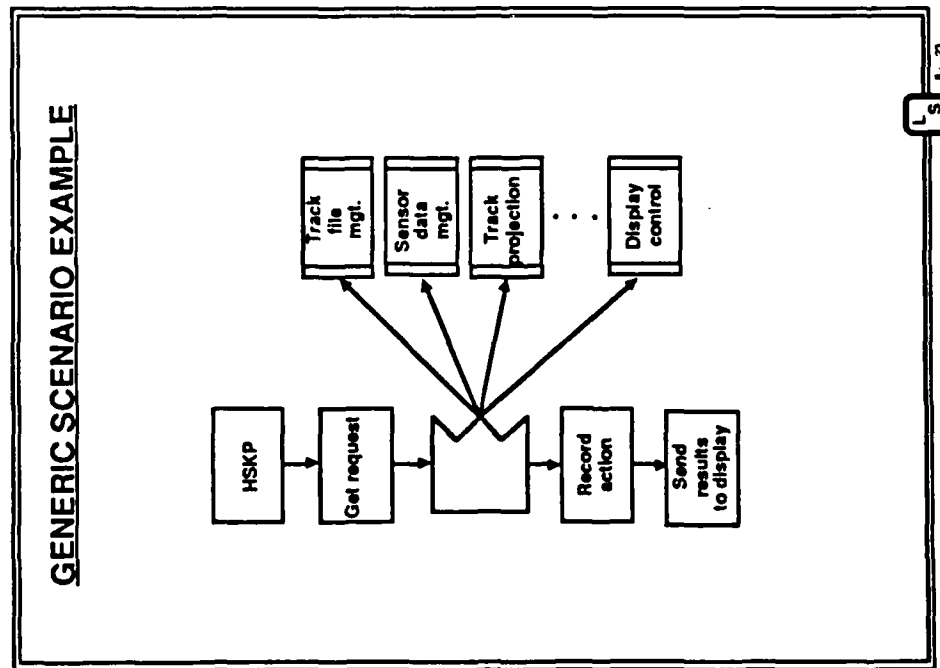


11 - 11

Notes

Ls

Software Performance Models

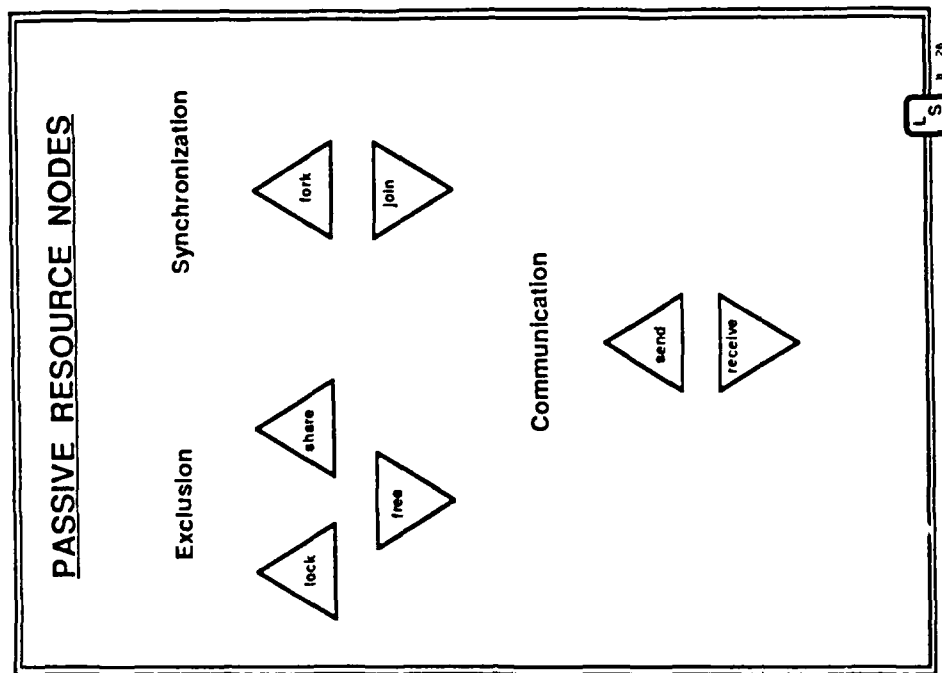
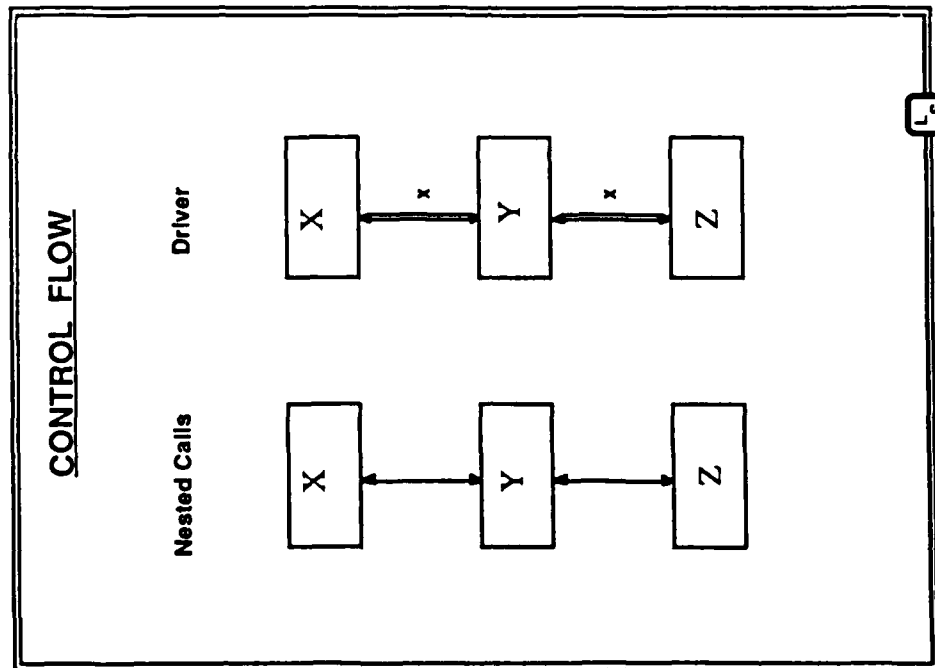


11 - 12

Notes

15

Software Performance Models

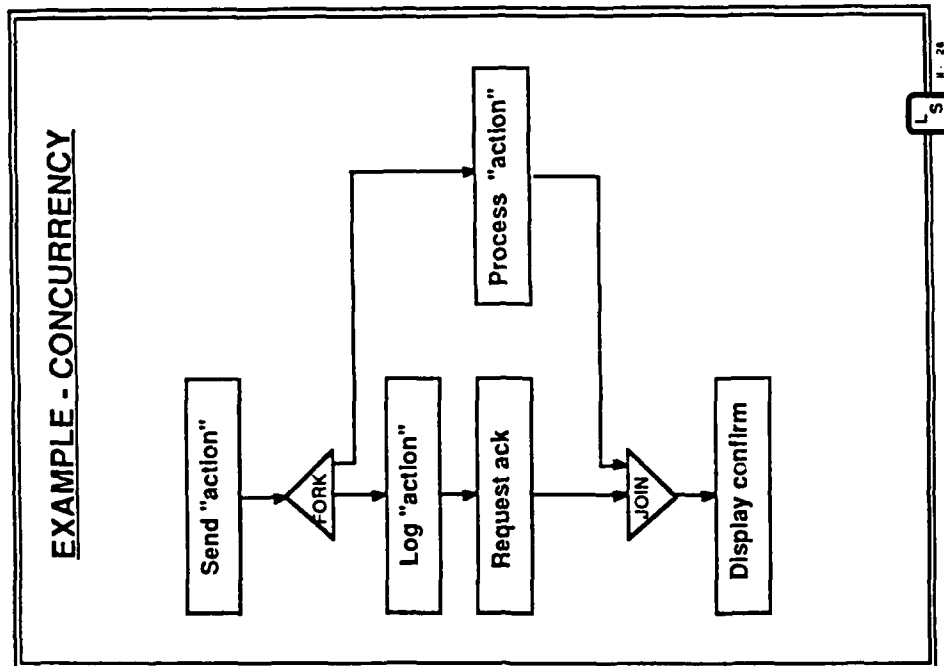
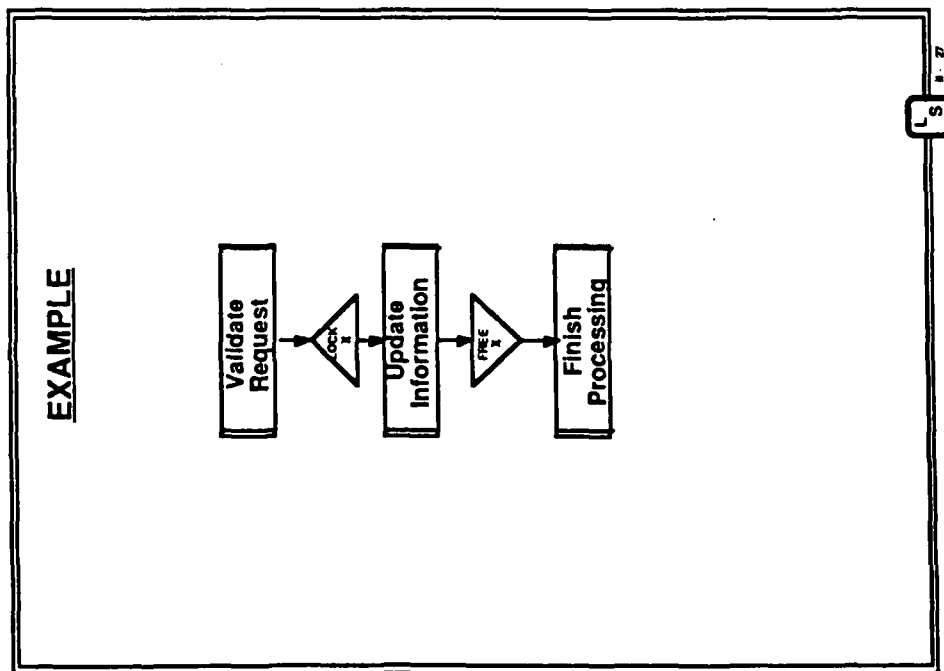


11 - 13

Notes

LS

Software Performance Models



EXERCISE: EXECUTION GRAPHS

Construct an execution graph for the signal processing scenario described below.

The **signal processing scenario** receives a frame consisting of 512*512 pixels. It executes *setup processing* to register and calibrate the frame. Next the scenario executes *filter processing* to remove clutter and identify (up to 1000) tracks. Finally, the scenario executes track processing to identify (up to 50) objects of interest from among the tracks.

Setup processing:

Setup processing executes (expanded) component *Frame_reg* then executes (expanded) component *Radiometric_corr*. *Frame_reg* performs a 2D interpolation which consists of a loop executed once per pixel that executes several arithmetic computations*. *Radiometric_corr* performs a 1D interpolation which consists of a loop executed once per pixel that executes several arithmetic computations.

Filter processing:

Filter processing executes (expanded) component *spatial_filter* then executes (expanded) component *temporal_filter*. *Spatial_filter* performs a 2D convolution which consists of a loop executed once per pixel that executes an inner loop (9 times) consisting of several arithmetic computations to "smooth" pixel intensity based on the intensity of the 9 adjacent pixels. *Temporal_filter* performs a third-order differencing which consists of a loop executed once per pixel that performs two steps: first it executes several arithmetic computations to "weight" pixels based on their intensity in previous frames, then it executes arithmetic computations to select pixels whose "weight" exceeds a threshold.

Track processing:

Track processing executes (expanded) component *fading_track* then executes (expanded) component *hough_transform*, then executes (expanded) component *candidate_selection*.

Fading_track performs a third-order differencing which consists of a loop executed once per track consisting of several arithmetic computations to eliminate fading tracks.

Hough_transform calculates track parameters by executing a loop (once per track) with an inner loop executed once per object. The inner loop performs two steps: first it executes several arithmetic computations to calculate "rho", then executes instructions to update a rho table.

Component *Candidate_selection* executes a loop (once for each of the 100 bytes of information stored for each of the objects) consisting of several computations to compute a threshold value.

* The arithmetic and other computations will be specified later.

Notes

LS

Software Performance Models

Part III

Software Models: Evaluation

L S II - 29

GRAPH ANALYSIS

- ◆ Best case
- ◆ Worst case
- ◆ Average
- ◆ Variance
- ◆ Distribution

L S II - 30

II - 15

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

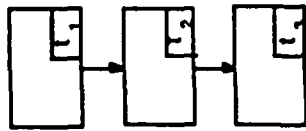
Notes

L'S

Software Performance Models

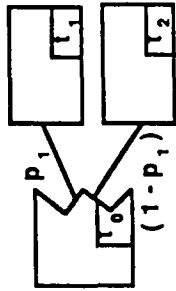
REDUCTION RULES AVERAGE ANALYSIS

Sequential



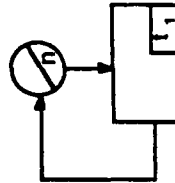
$$T = t_1 + t_2 + t_3$$

Conditional



$$T = t_0 + p_1 t_1 + (1 - p_1) t_2$$

Repetition



$$T = n t_1$$

L S 31

OTHER ANALYSES

Expanded Nodes

Control Flow

Best, worst

Variance, distributions

Checkpoint evaluation -
elapsed time without contention

*Optimistic evaluation - if it doesn't
meet performance goal, fix problems
before more realistic evaluation*

Example: Timing check of
new algorithm:

If OK, check impact on other timing loops

If not, examine other algorithms

L S 32

II - 16

Notes

LS

Software Performance Models

PASSIVE RESOURCE NODES

Exclusion
Lock, free
Synchronization
Fork, join
Communication
Send, receive

Best / Worst
Cases

EXERCISE

The following chart specifies the instructions executed in the signal processing scenario. Compute the execution time for the Hough transform component.

Component	Fetch	Store	Compare	Add	Multiply
Frame_reg	6	1	1	3	1
Radiometric	3	1	0	1	1
Spatial_filter	2	1	0	1	1
Temporal_filter	8	1	0	3	4
+ threshold	1	1	0	1	0
Fading_track	8	0	0	3	4
Hough	4	0	0	1	2
+ update	0	1	0	1	0
Candidate_sel	1	1	0	1	0

CE execution times (microsecs)

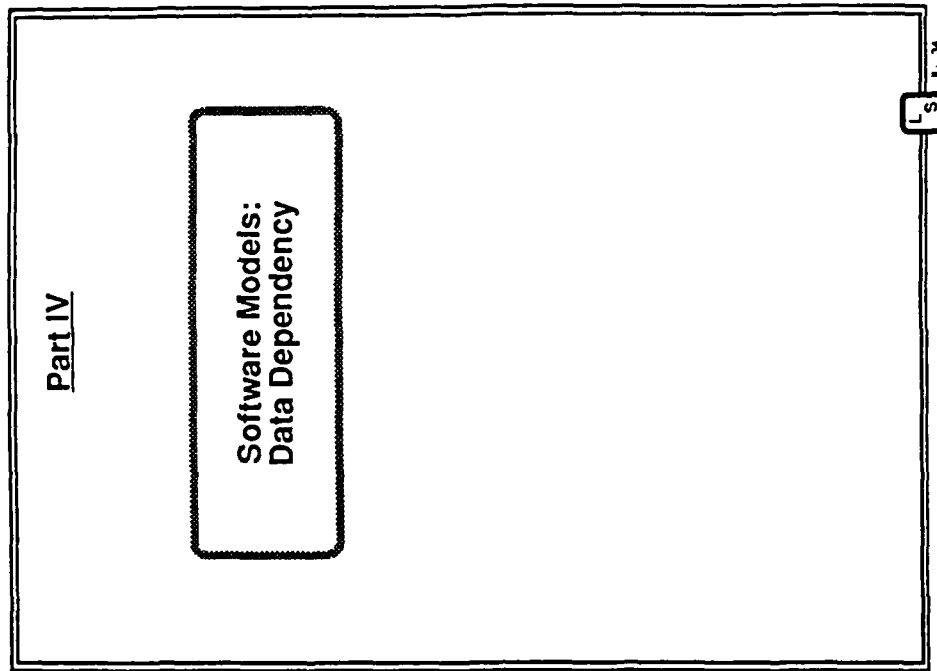
Fetch	20
Store	20
Compare	50
Add	50
Multiply	200

11 - 17

Notes

L S

Software Performance Models



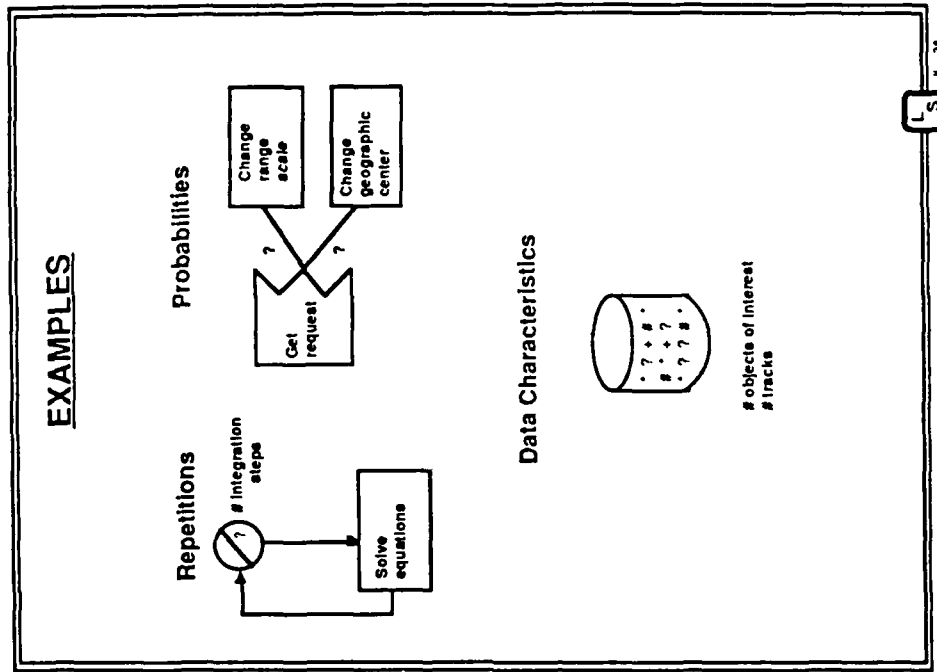
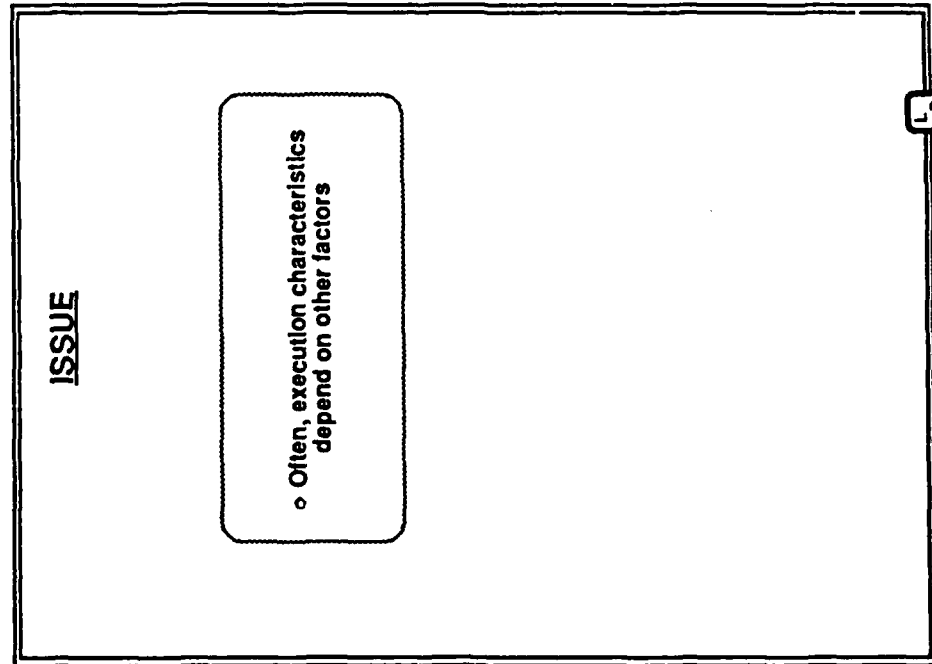
II - 17

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

LS

Software Performance Models



II - 18

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

LS

Software Performance Models

SOLUTIONS

Best case analysis
 Shortest path
 Worst case analysis
 Longest path
 More realistic
 "Typical" situations
 Scenario choice
 Define conditions of Interest
 Use parameters for data dependencies
 Specify variables rather than
 absolute resource requirements
 Solve for performance sensitivity to
 variable values

L S 8. 37

CONDITIONAL PERFORMANCE GOALS

Select conditional performance goals
 for various n

<u>Value</u>	<u>Goal</u>
$n \leq 50$	1 sec.
$50 < n \leq 200$	3 sec.
$200 < n \leq 500$	10 sec.

L S 8. 38

11 - 19

Notes

LS

Software Performance Models

Part V

Performance Tools

DATA COLLECTION TOOLS

Event Recorders
System level
Program level
Major subsystems
Monitors
Hardware
Software
Hybrid
Software Instrumentation

11 - 20

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

LS

Software Performance Models

MODELING TOOLS

Simulation
PAWS
SCERT
SLAM
SIMSCRIPTII
CSIM
SIMULA
GPSS
Analytic - system models
BEST/1
MAP
Others
POD
ADAS

L S R. 41

DEVELOPMENT TOOLS

L S R. 42

II - 21

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

15

Software Performance Models

OTHER INFORMATION SOURCES

Statement timings
Database call timings
Path length analysis
Trace data banks

L S R 43

SUMMARY

- > Data requirements, sources identified
- > Software models
Representation
Evaluation
Data dependency
- > Performance tools that support SPE

L S R 44

11 - 22

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Software Model Workshop

LS

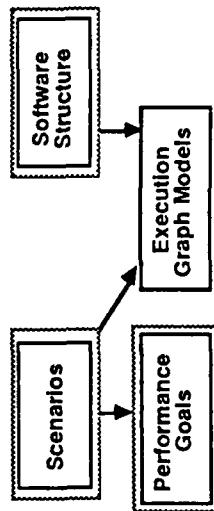
Software Model Workshop

OBJECTIVES

- To introduce the POD modeling tool
- To relate POD to the SPE methodology
- To use POD to evaluate software scenarios and alternatives

L S M. 1

SOFTWARE MODEL COMPONENTS



Scenarios - Representative workloads

*Performance goals - response times
throughput
resource budget*

Software structure - major components

*Execution models - components executed
order, frequency*

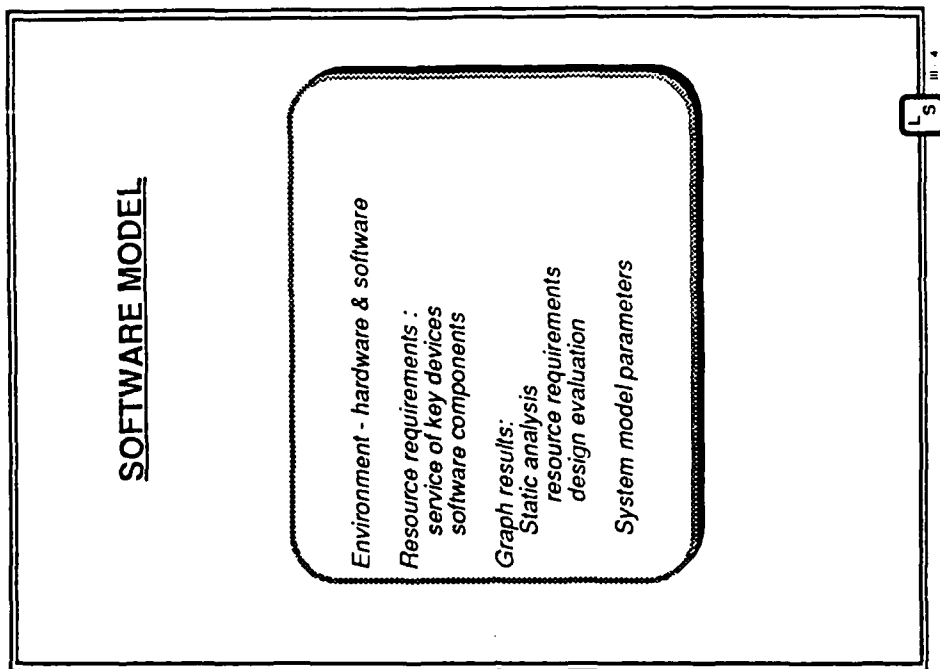
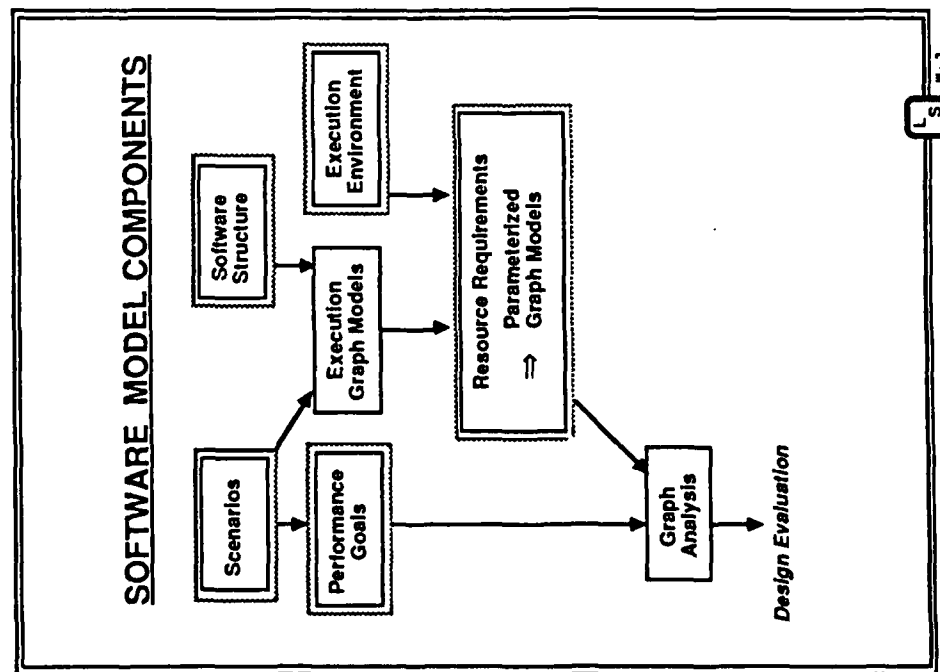
L S M. 2

III - 1

Notes

19

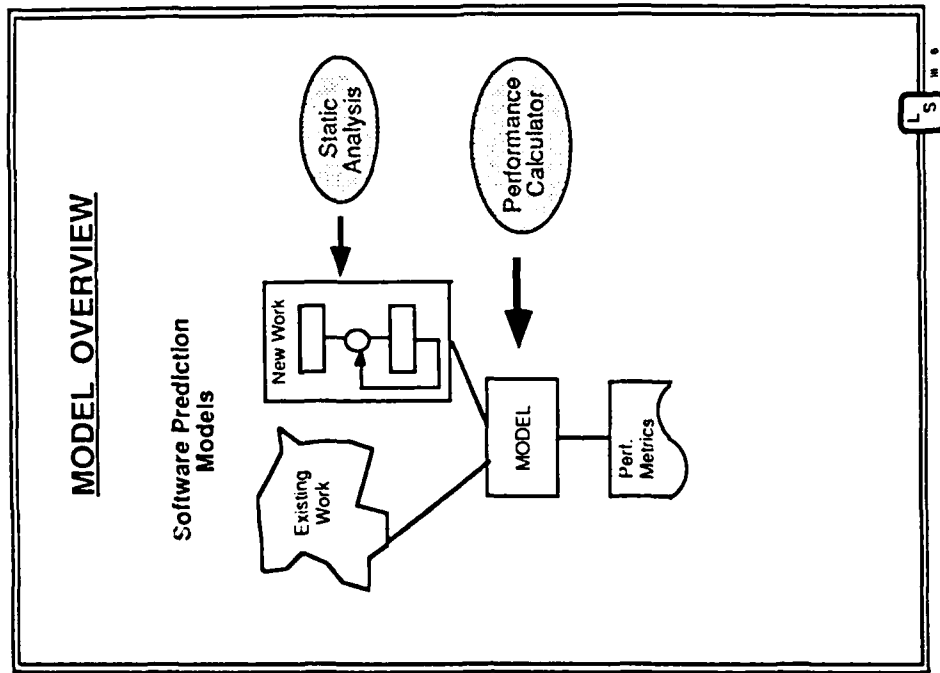
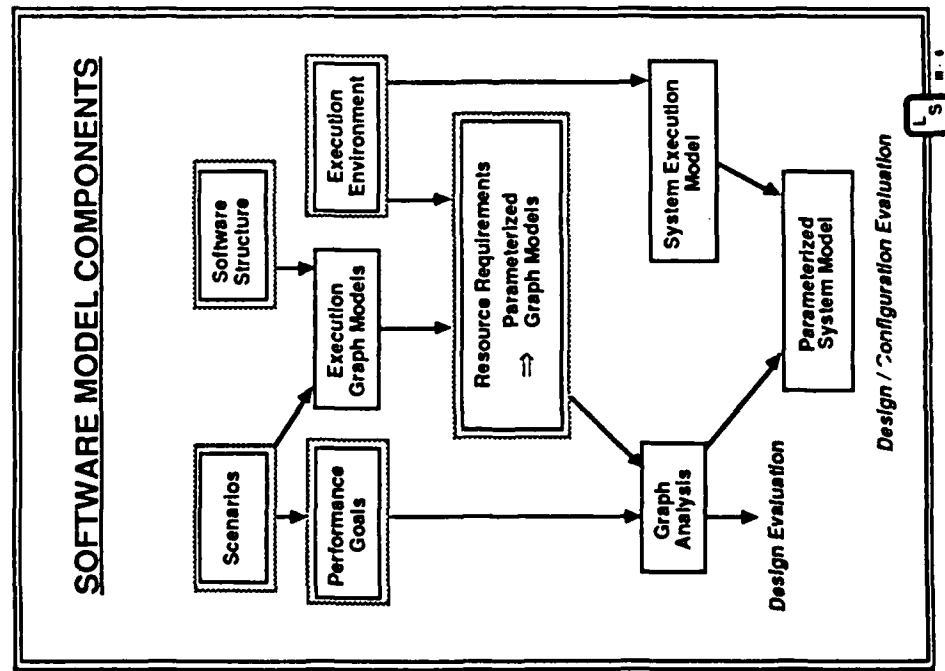
Software Model Workshop



Notes

L's

Software Model Workshop

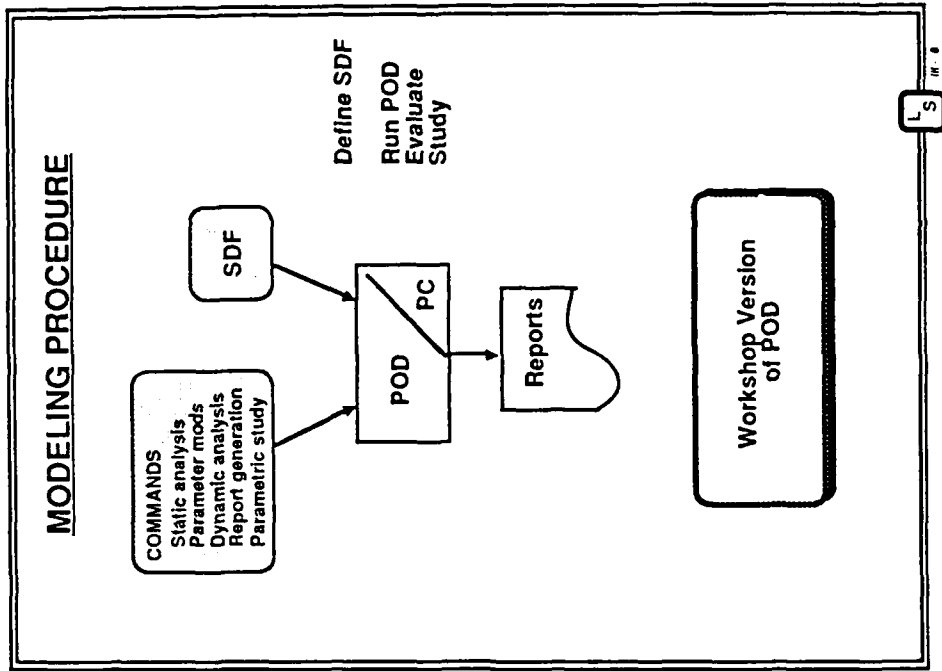
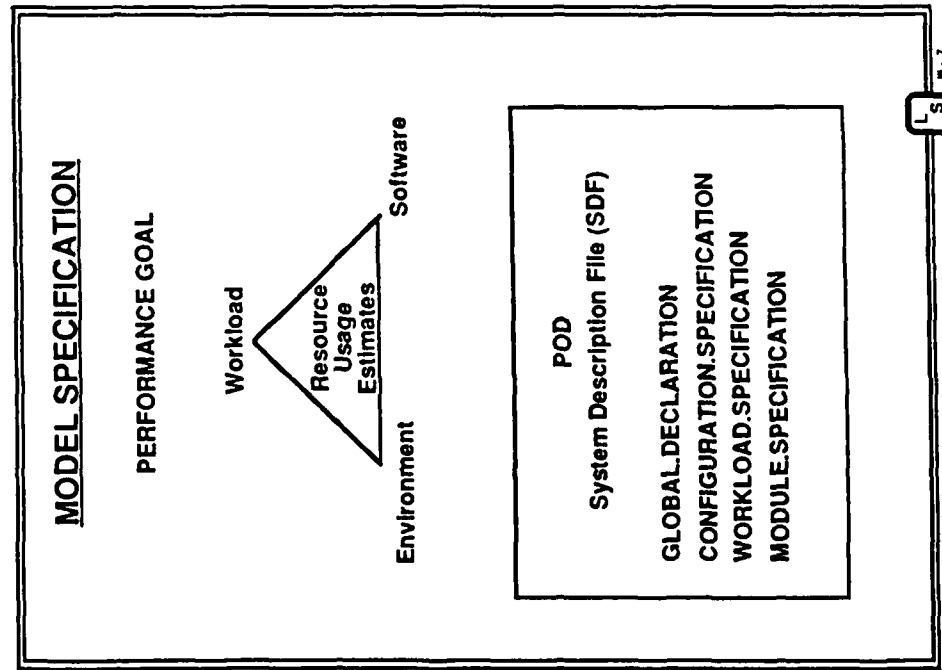


III - 3

Notes

L S

Software Model Workshop



III - 4

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

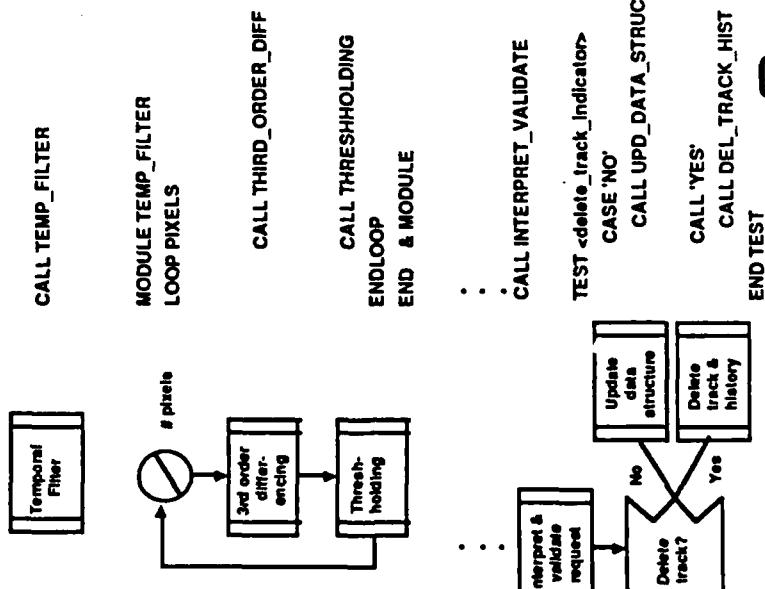
Notes

9

Software Model Workshop

MODULE SPECIFICATIONS

Define Execution Structure



RESOURCE REQUIREMENTS

Define software components' usage of computer system resources

Syntax:

[EST
ACTUAL] <device> USAGE = [<expression>
<parameter>] (<units>)

Example:

```

MODULE THRESHOLDING
EST CPU_0  USAGE =      & fetch
      .000020
      + .000050      & add
      + .000020 MSEC  & store
END
    
```

III - 5

Notes

LS

Software Model Workshop

CONFIGURATION SPECIFICATIONS

Define computer system devices and service rates

Syntax:

```

DEVICE  <device name>
TYPE = <device type>
<device attributes>
END

```

Examples:

```

DEVICE  CPU_0
TYPE = CPU
RATE = 1 MIPS
END

```

```

DEVICE  CPU_1
TYPE = DEFINED_CPU
OPERATION  I_FETCH
TIME = .000020 MSEC
OPERATION  I_STORE
TIME = .000020 MSEC
.
.
.
END

```

```

MODULE THRESHOLDING
EST CPU_1 USAGE =
1 I_FETCH + 1 I_ADD + 1 I_STORE
END

```

L S 11

WORKLOAD SPECIFICATION

Define:

```

Scenario names
Software Invoked
Workload Intensity      (for dynamic analysis)

```

Syntax:

```

WORKLOAD  <name>
TYPE = <workload type>
<workload attributes>
END

```

Example:

```

WORKLOAD SPECIFICATION
WORKLOAD  ONLY_ONE
TYPE = PERIODIC
JOB_STREAM = SIG_PROC
ARRIVAL_RATE = 1 * 3600
& 1 frame per second
& specified in jobs per hour
END
END

```

L S 12

III - 6

Notes

L S

Software Model Workshop

STATIC ANALYSIS

1. Create SDF
 2. Invoke POD
 3. READ <sdf file name>
 4. Produce reports:
 - TIMES & Total usage in msec.
 - & for each device in each workload
 - MODSTAT <job stream name> <device>
 - & Per call usage of specified device
 - & Number of calls per job
 - & For each module of the specified job
 - PATHTIMES <job stream name> <device>
 - & Case tree for job and
 - & Maximum usage times for each case
 - MAXPATH <job stream name> <device>
 - & Longest path
 - JOBPRO <job stream name> <device>
 - ALL ALL
 - & Number of operations for device,
 - & Time per operation (ms.)
 - & Total for all operations (ms.)
 - & Percent of total service time
- (etc.)

L S M 13

WORKSHOP EXERCISE I

Signal processing case study

1. Edit the SDF named sig_proc. It is a template that specifies most of the sdf data for the earlier signal processing case study. Refer to the specifications in section II.
2. Insert execution times for the operations for device CPU_1.
3. Specify the missing execution steps and resource requirements for the TRACK_INIT module.
4. Solve the model. Fill in the first column of results on the next page.
5. Does the scenario meet its performance objectives? List 2 options for improving performance:
 - (1)
 - (2)
6. Change the number of pixels to 1024 * 1024 and re-solve the model. Fill in the second column of results on the next page.
7. Change the time for the multiplication operation to .00075 msec. and re-solve the model. Fill in the third column of results on the next page.

L S M 14

III - 7

EXERCISE RESULTS					
STATISTICS	Original Specs	More pixels	Faster Multiply	Distributed processing	(optional) New algorithm
-----	-----	-----	-----	-----	-----
Response time					

CE Residence Time:					
Total					
Setup_proc					
Frame_reg					
Radiometric					
Filter					
Spatial					
Temporal					
Track_init					
Fading_track					
Hough_transform					
Candidate_sel					

CE Throughput:					
Total					
Frame_reg					
Radiometric					
Spatial filter					
Temporal filter					
Fading_track					
Hough_transform					
Candidate_sel					

CE Utilization:					
Total					
Frame_reg					
Radiometric					
Spatial filter					
Temporal filter					
Fading_track					
Hough_transform					
Candidate_sel					

Notes

L S

Software Model Workshop

WORKSHOP EXERCISE II

Revised signal processing case study

1. Edit the SDF named `distr_sig_proc`. It is a template for an 8 processor distributed system. Each stage of the process will execute on a different processor. Frames will be stored in global memory with a 100 nsec. (.0001000 msec) access time.
2. Insert the execution times for the global memory fetch and store.
3. Assign steps in the process to different CPU's (Note - Assign all operations within a loop to the same processor).
4. Solve the model. Fill in the fourth column of results.
5. Briefly describe your evaluation of this alternative. Suggest an alternative distributed processing strategy.
6. (optional) Replace one of these signal-processing algorithms with an alternative. Change the resource specifications to reflect your algorithm and re-solve the model. Compare your results.



III - 8

Role of Static Analysis

L
S

Role of Static Analysis

OBJECTIVES

- > To review the purpose of models for SPE
- > To examine typical static analyses
- > To identify situations that require dynamic analysis

L S N. 1

SPE GOALS

- ◆ Supplement existing methods
- ◆ Ease of use
- ◆ Rapid assessments
- ◆ Sufficient precision for purpose of analysis
- ◆ Match modeling effort to data precision
- ◆ Life cycle usefulness

L S N. 2

IV - 1

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

This is a full-page view of a sheet of graph paper. The page is covered by a uniform grid of small squares formed by dotted lines. A thick black border frames the entire page. In the top right corner, there is a small rectangular tab containing the letter 'L' followed by a subscripted 'e' (Le).

Role of Static Analysis

MODEL REQUIREMENTS

GOAL:

Initially

- Distinguish between "good" and "bad"

Later

- Increase precision of predictions
- Provide implementation decision support
- Identify problem areas
- Provide guidance for testing

L S N: 3

ANALYSIS STRATEGY

Resource Requirements
Expected Value & Upper Bound

Identify Critical Components
Prototype
Implement First

Simple,
Optimistic



More Complex,
Realistic

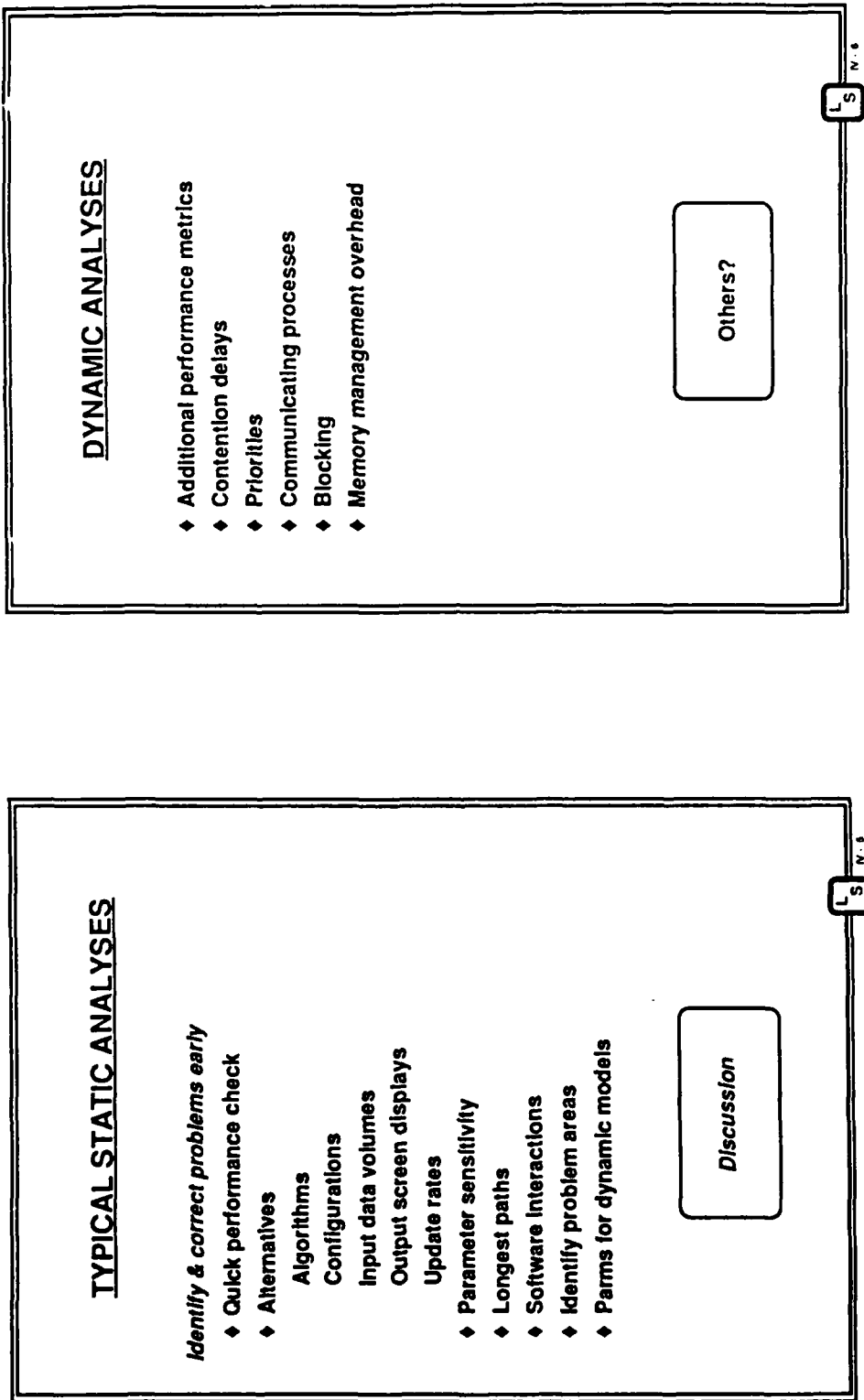
L S N: 4

IV - 2

Notes

LS

Role of Static Analysis



IV - 3

Notes

LS

Role of Static Analysis

MODEL FIDELITY

- ◆ Queueing Network Models:
Utilizations within 5 - 10%
Residence and Response Times within 30%

- ◆ Software Models:
Within 1% when measurements provide
model parms

Precision depends on specifications

Experience:

Performance Problem Predictions Accurate
CPU & I/O Estimates within Bounds
Global memory estimates slightly low

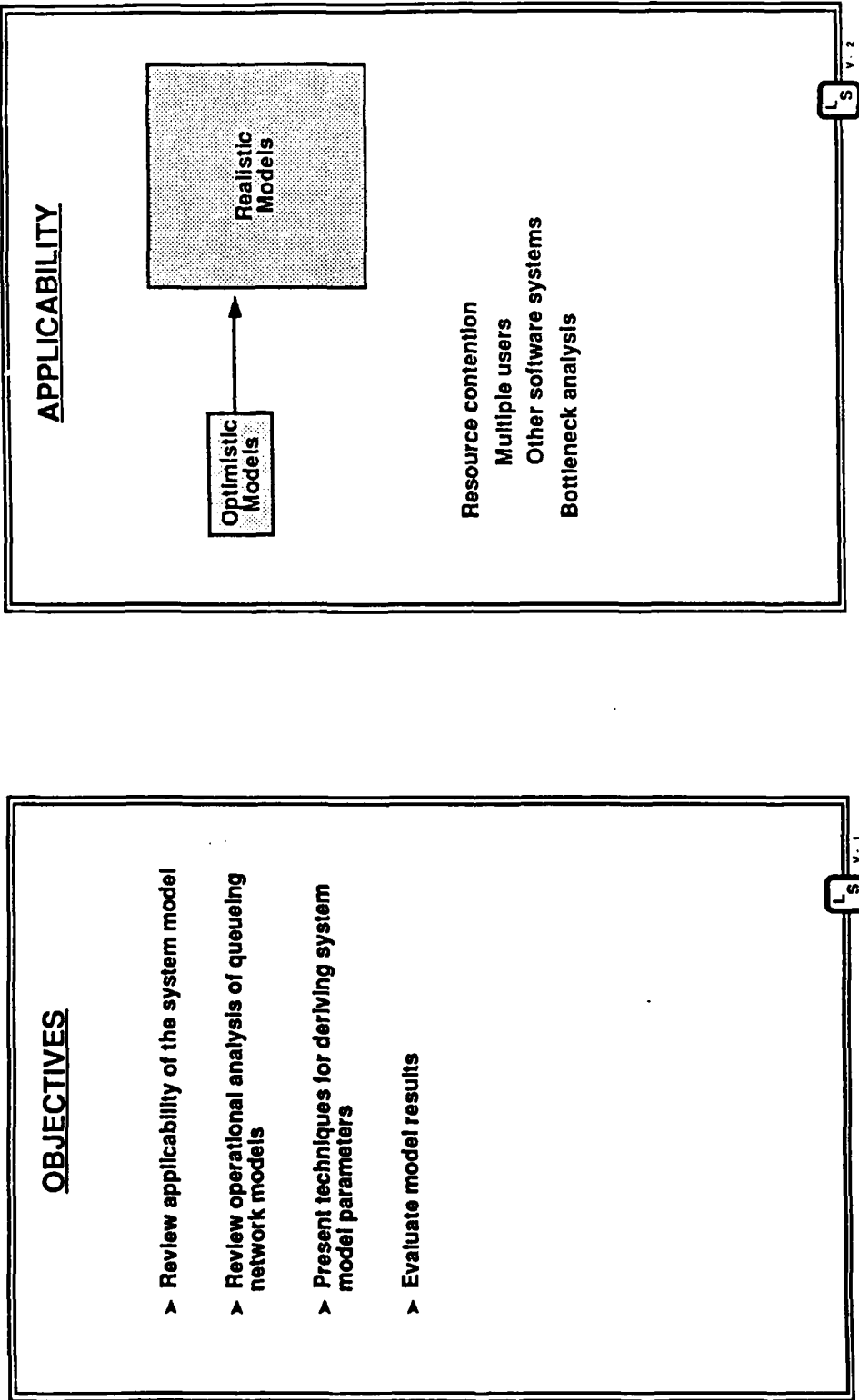


IV - 4

System Execution Models

LS

System Execution Models

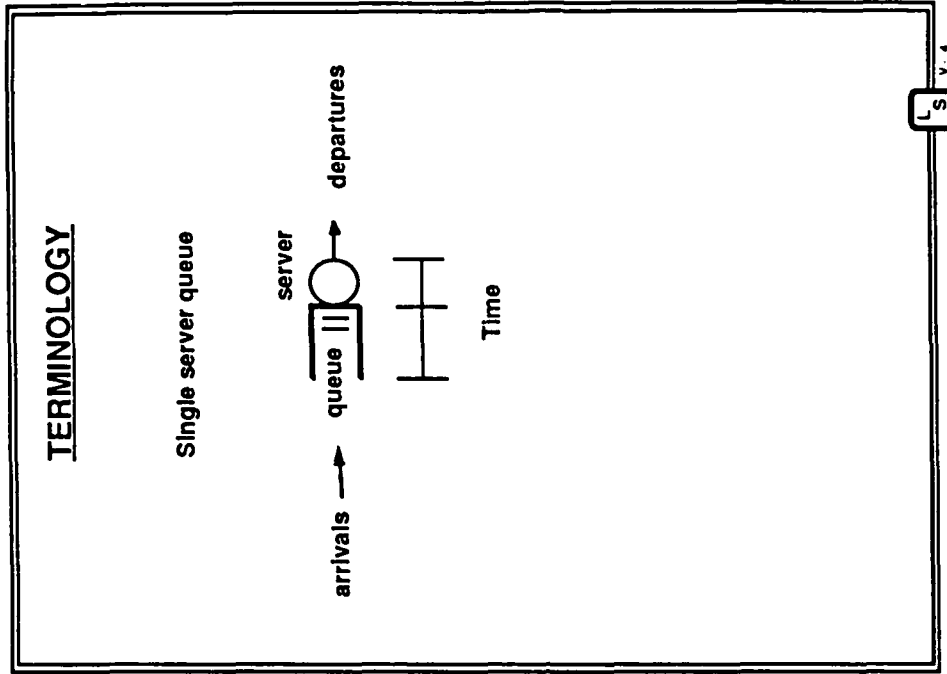
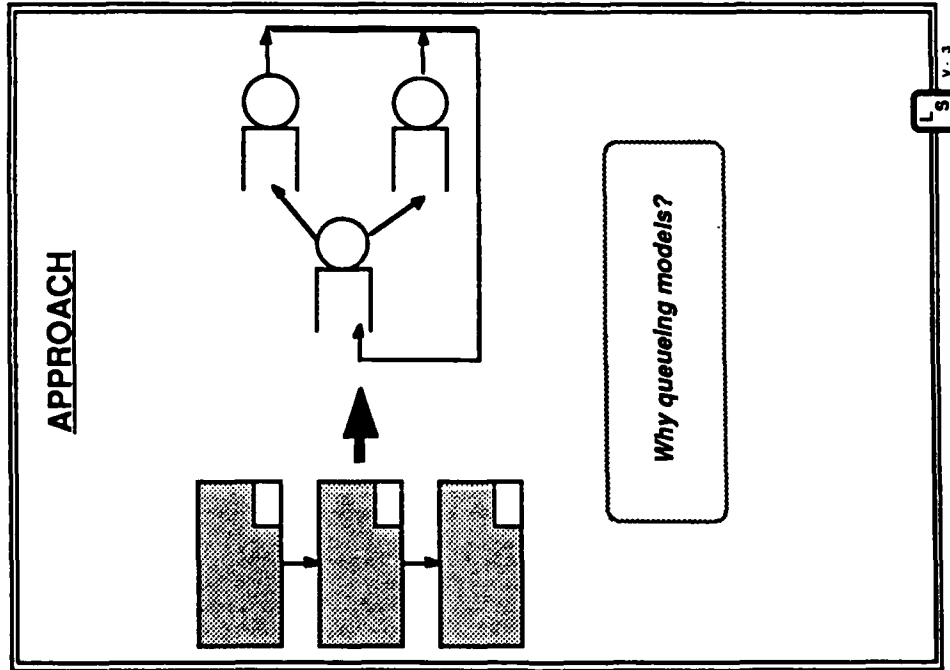


V - 1

Notes

15

System Execution Models



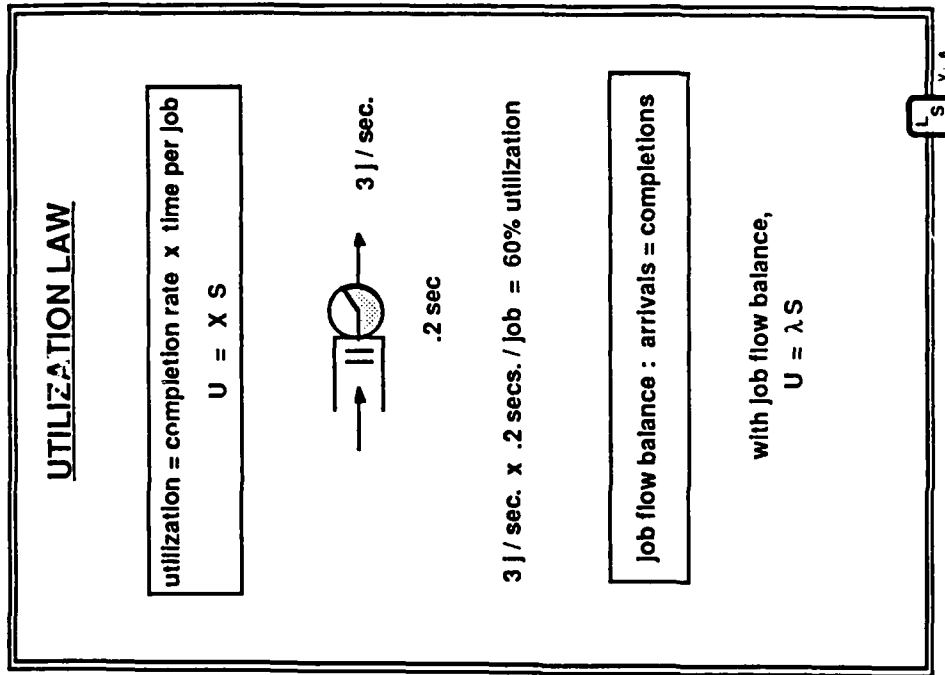
V - 2

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

15

System Execution Models



V - 3

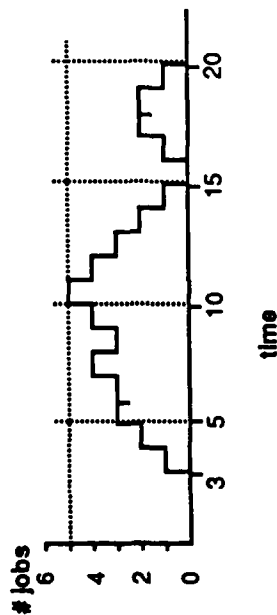
© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

15

System Execution Models

COMPUTATIONS



Complete the following:

Measurement period, T _____

Arrivals, A _____

Completions, C _____

Busy Time, B _____

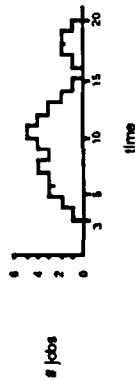
Throughput, $X = C / T$ _____

Utilization, $U = B / T$ _____

Mean service time, $S = B / C$ _____

L S V. 7

LITTLE'S LAW



W = area under graph
= job seconds (1 jobs for 3 seconds)

mean queue length, \bar{n} = average height

$$\bar{n} = \frac{W}{T}$$

average time per completed request, R

$$R = \frac{W}{C}$$

Little's Law:

mean queue length = throughput \times response time

$$\bar{n} = X R$$

L S V. 8

V - 4

Notes



System Execution Models

APPLICATIONS

Metric calculation

Calculate metrics from measurements

Verification

Ensure that metrics are consistent.

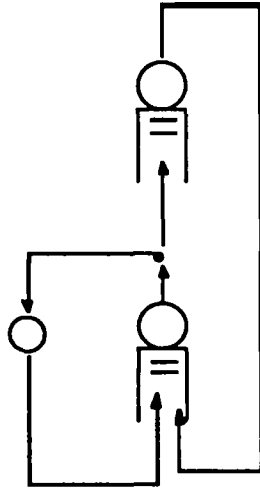
Check measurements, calculations, model assumptions

Performance prediction

Forecast changes in key metrics and calculate predicted performance indicators

L S V. 9

QUEUEING NETWORK MODELS



A central server model

- ♦ Jobs at one device at a time
- ♦ Servers are busy if a job is in the queue

L S V. 10

V - 5

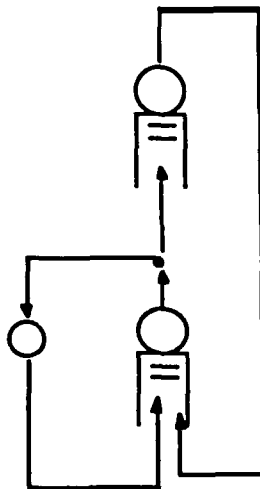
© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

19

System Execution Models

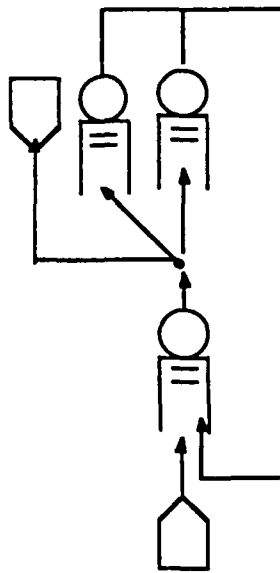
CLOSED MODEL



- ◆ Specify number of users (or operator consoles) and 'think time'
- ◆ Throughput and response time depend on number of users

L S V. 11

OPEN MODEL



- ◆ Specify arrival rate
- ◆ Rate is independent of response time and number of jobs in the system

L S V. 12

V - 6

Notes



System Execution Models

APPLICATION

Metrics	System	CPU	DISK1	DISK2
Arrival rate, X_0	5 j/sec.			
Visits, V_i	1	3	3	1
Tput, X_i				
Service, S_i		.03	.025	.1
Util., U_i				
Residence, R_i				
Queue length, \bar{n}_i				

FORMULAS

dev. tput = sys. tput x dev. visits

$X_i = X_0 V_i$

dev. util. = dev. tput x dev. serv. time

$U_i = X_i S_i$

dev. res. time = dev. serv. / (1 - dev. util.)

$R_i = \frac{S_i}{1 - U_i}$

dev. queue len. = dev. tput x dev. res. time

$\bar{n}_i = X_i R_i$

V - 7

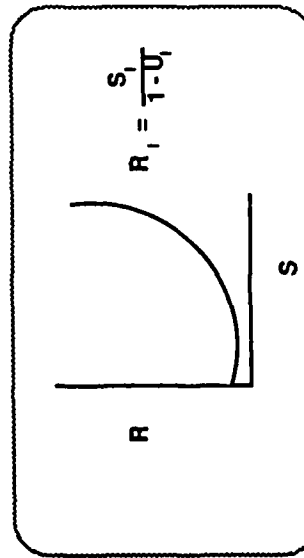
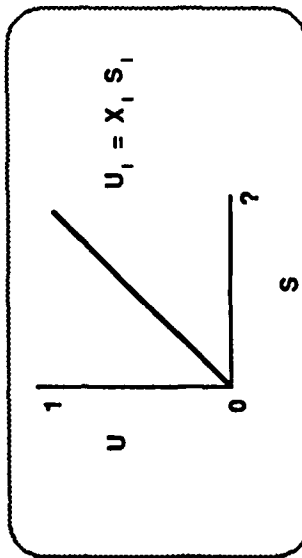
Notes

15

System Execution Models

SERVICE TIME IMPACT

When $X_i = 1$



L_S V. 18

SYSTEM RESPONSE TIME

Little's Law

$$R = \frac{N}{X_0}$$

Response time = Number in system / Tput

♦ Number in system, N = sum of dev. queue lengths, $N = \sum_i n_i$

> Compute N and R for problem VI-13

Note from Little's Law,

$$R = \sum_i V_i R_i$$

L_S V. 18

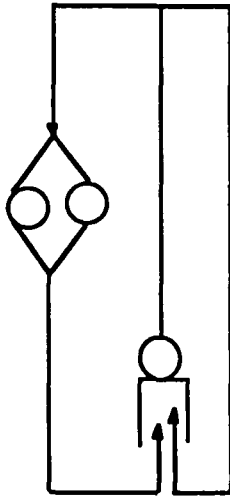
V - 8

Notes

15

System Execution Models

CLOSED SYSTEM RESPONSE TIME



Workload Intensity:
Number of users, M
Think time, Z

Interactive Response Time Formula

$$R = \frac{M}{X_0} + Z$$

EXAMPLE

- ◆ There are 24 operators
- ◆ Think time is 20 seconds
- ◆ Each job makes 20 disk requests (visits)
- ◆ Disk utilization is 40%
- ◆ Mean disk service time is 25 ms.
- ◆ What is the average response time?

$$R = \frac{24}{X_0} + 20$$

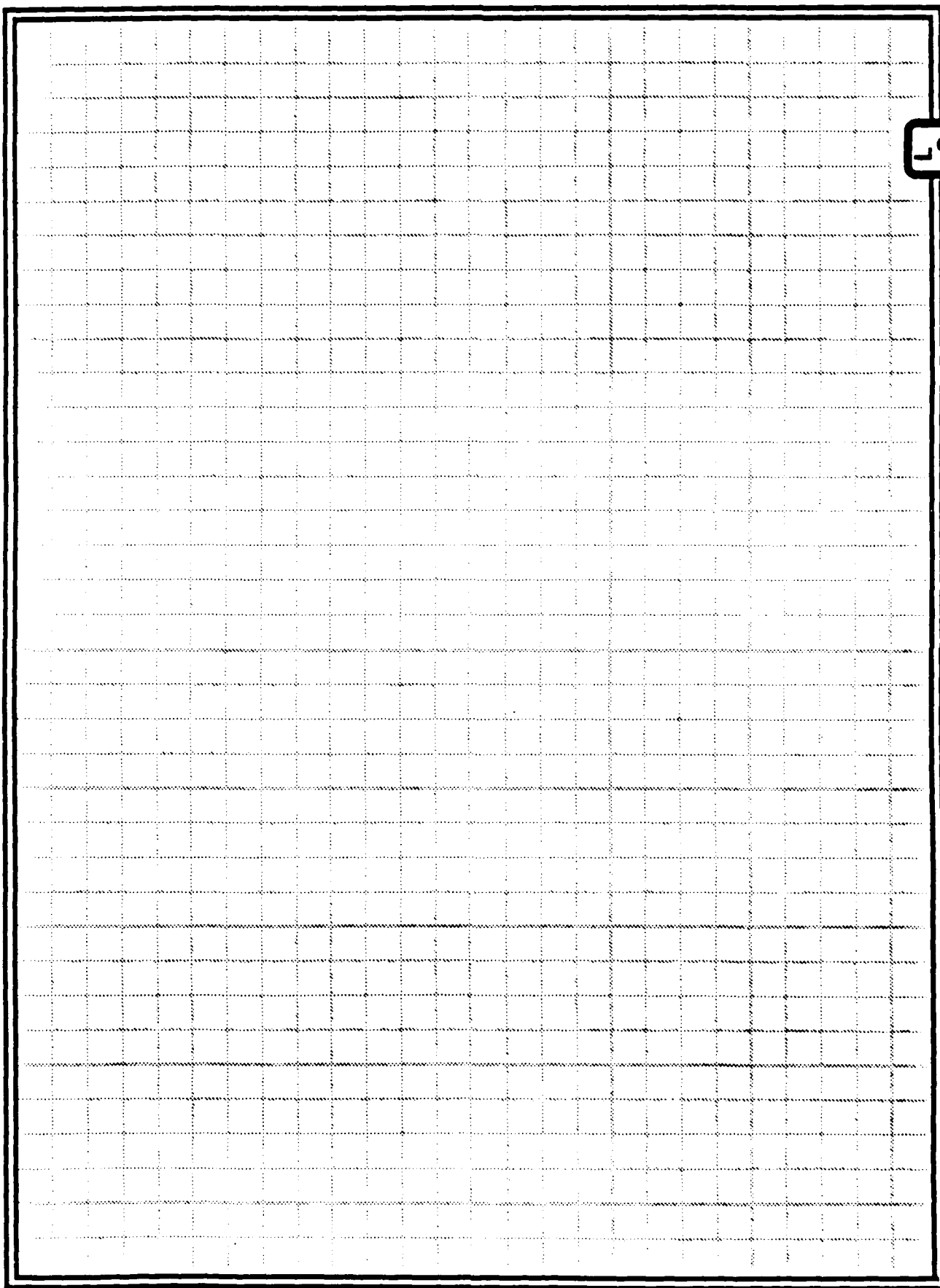
$$X_1 = X_0 V_1 \quad \Rightarrow \quad X_1 = X_0 (20)$$

$$U_1 = X_1 S_1 \quad \Rightarrow \quad .4 = X_1 (.025)$$

$$R = \underline{\hspace{2cm}}$$

V - 9

Notes



15

System Execution Models

ILLUSTRATION:
REVISE MODEL PARAMETERS

What if the disk service time is reduced to 3 ms.
by using cached disks?

$$R = \frac{24}{X_0} + 20$$

$$X_1 = X_0 V_1 \quad \rightarrow \quad X_1 = X_0 (20)$$

$$U_1 = X_1 S_1 \quad \rightarrow \quad .4 = X_1 (.003)$$

$$R = \underline{\hspace{2cm}}$$

What went wrong?

Be careful of invariance assumptions!

L S

 V. 19

Part II

**Model
Parameters**

L S

 V. 20

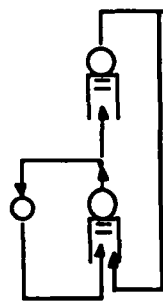
V - 10

Notes

15

System Execution Models

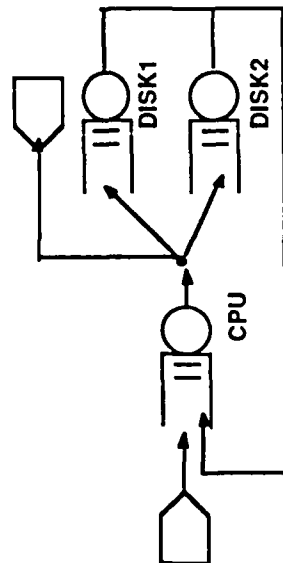
DATA REQUIRED



- ♦ Mean service time for each device
- ♦ Visits to each device
- ♦ Workload Intensity
 - Open models - arrival rate
 - Closed models - number of users
- think time

L S V 21

EXAMPLE



- ♦ Jobs arrive from the source at rate 5 jobs / sec.
- ♦ Each job requires an average of
 - 90 ms. CPU time (total)
 - 3 I/O's at DISK1
 - 1 I/O at DISK2
- ♦ Average service time
 - DISK1 is 30 ms.
 - DISK2 is 60 ms.

- ♦ Calculate remaining model parameters

Why would disk service times differ?

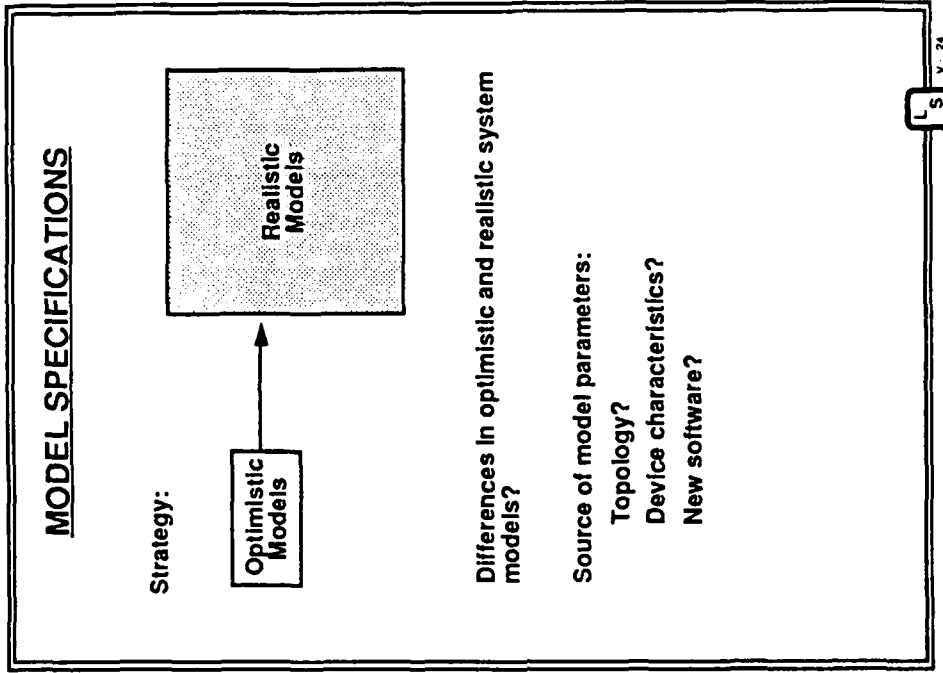
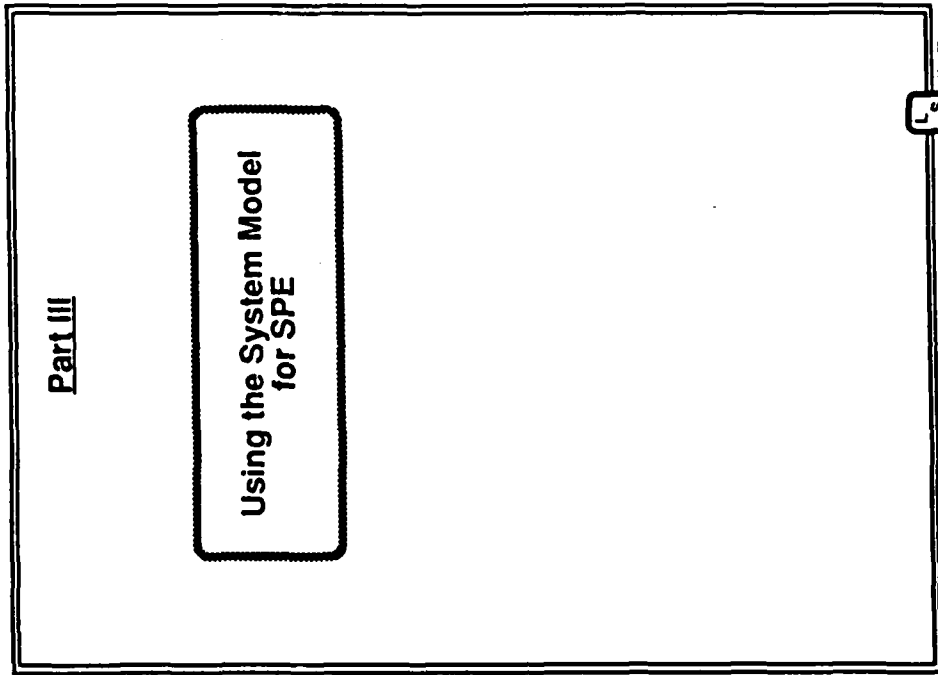
L S V 22

V - 11

Notes

15

System Execution Models

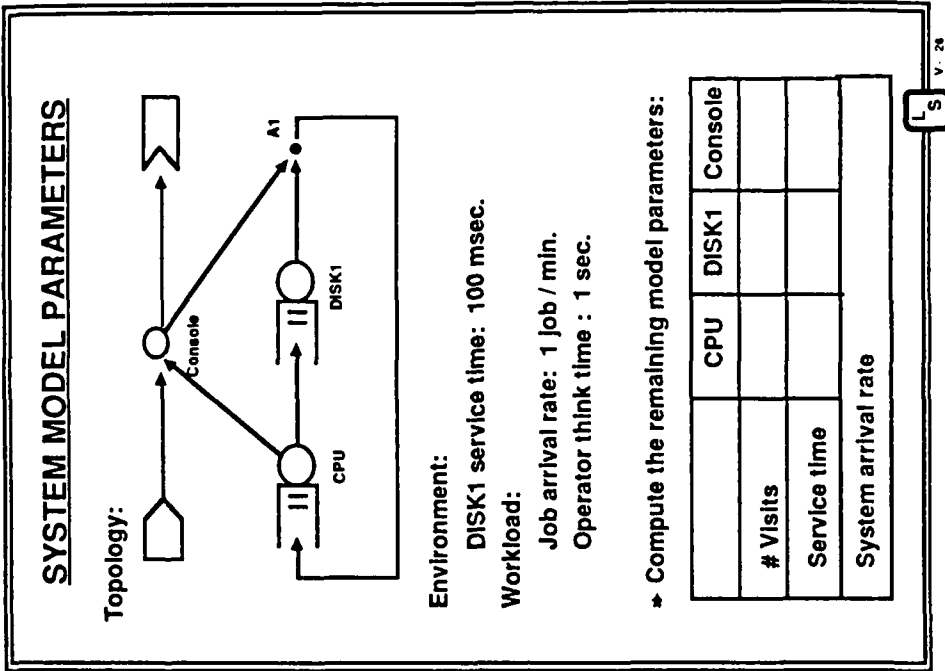
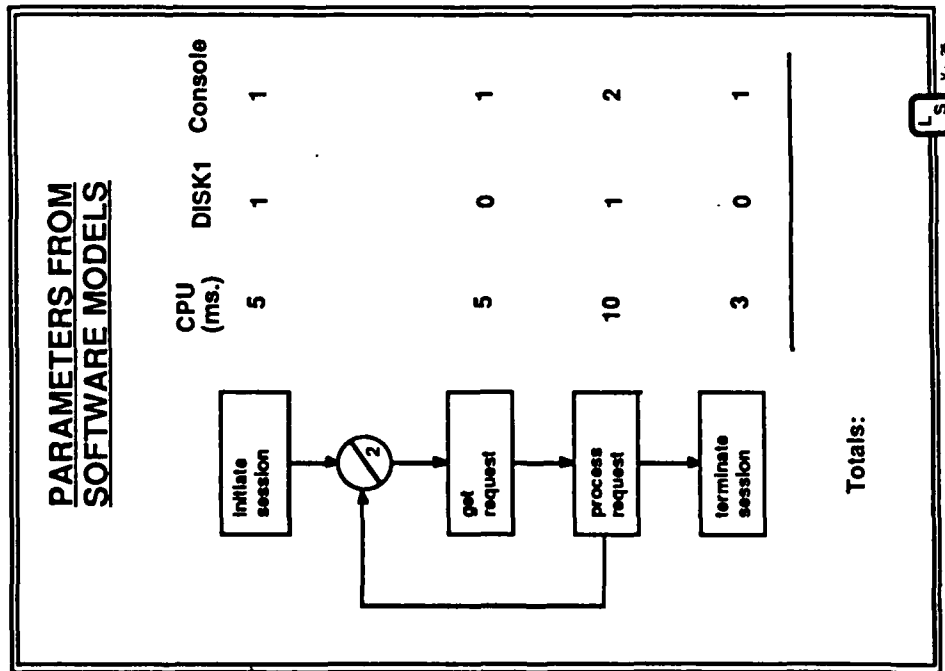


V - 12

Notes

L's

System Execution Models



V - 13

Notes

LS

System Execution Models

DISCUSSION

- ♦ What is the minimum response time?
- ♦ How are other workloads included?
- ♦ Note that the CPU times are specified in the software model. What if the CPU has not been selected?
- ♦ How can you determine the maximum arrival rate that the system will support?

L S V. 27

MODEL EVALUATION

Bottleneck device, b:
 $\max(V_i S_i)$

Maximum throughput:
 $\max X_0 = \frac{1}{V_b S_b}$

(Derivation)

From utilization law, max utilization is 1 so

$$U_b = X_b S_b \quad \text{and} \quad X_b = V_b X_0$$
$$1 = V_b X_0 S_b$$

L S V. 28

V - 14

Notes

LS

System Execution Models

BOTTLENECKS

- ◆ What is the bottleneck in Example V-13?
- ◆ What is its maximum throughput?
- ◆ What are some strategies for improving a system with a bottleneck?

L S V. 28

SUMMARY

- Reviewed operational analysis fundamentals
- Discussed required model parameters
- Used the system execution model for SPE

L S V. 30

V - 15

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

System Model Workshop

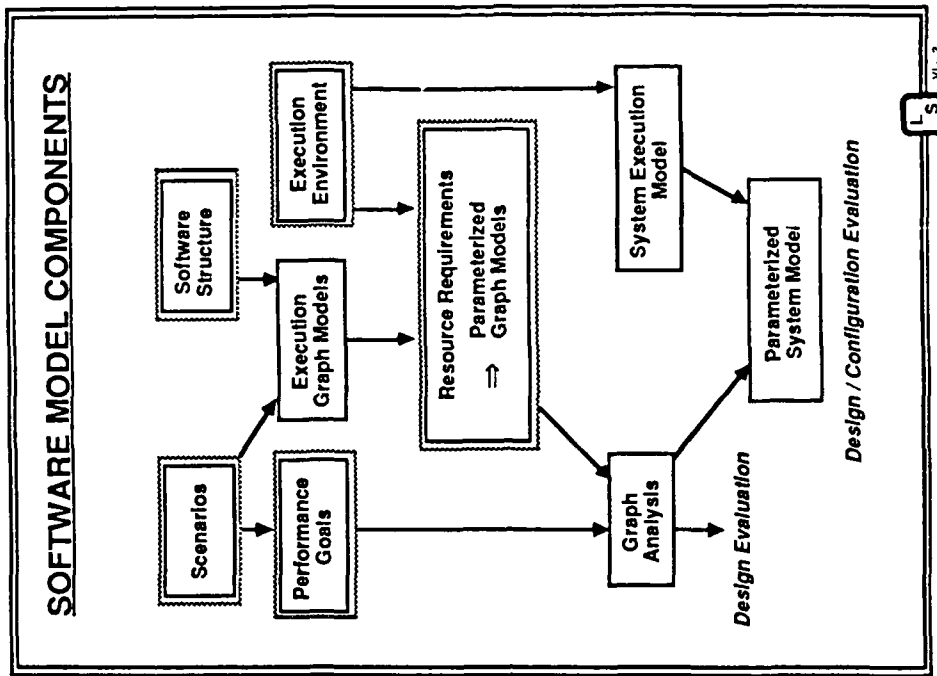
15

System Model Workshop

OBJECTIVES

- > Review the role of system models in the SPE methodology
- > Introduce POD's Performance Calculator (PC)
- > Use POD to evaluate system models
- > Solve more realistic case studies

L S VI - 1

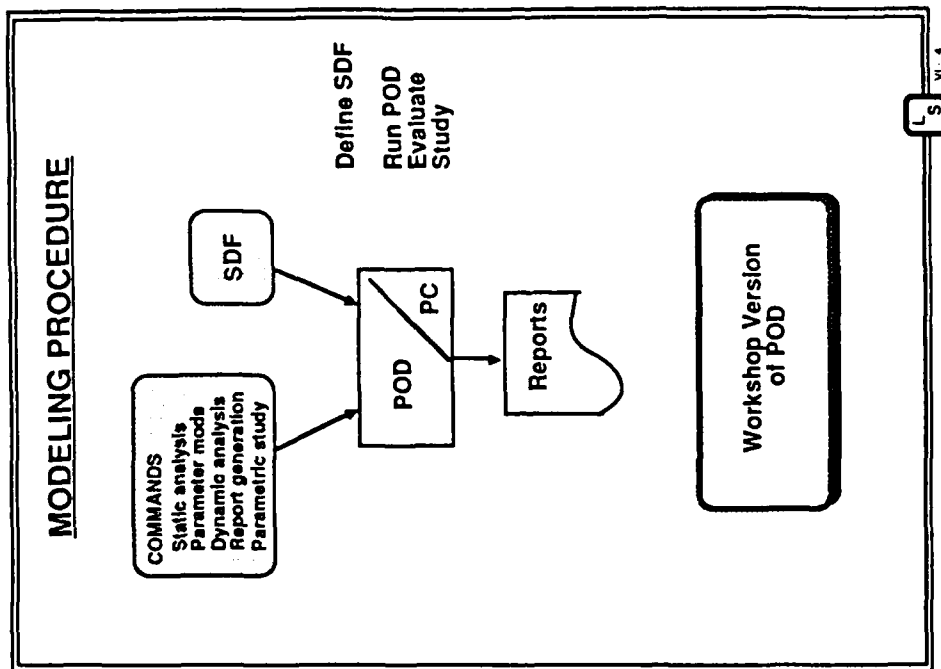
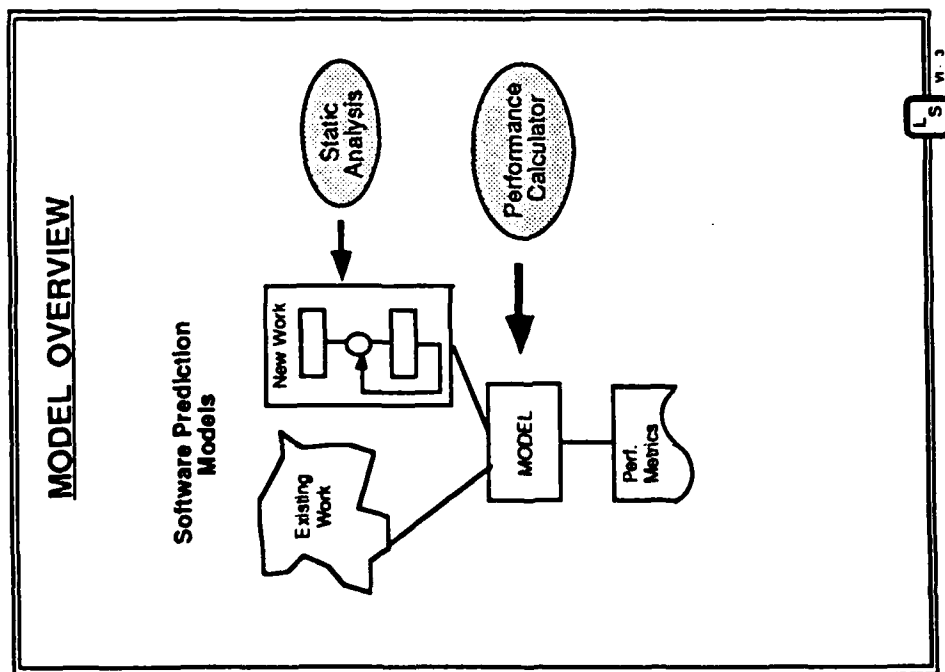


VI - 1

Notes

LS

System Model Workshop



VI - 2

Notes

LS

System Model Workshop

CONFIGURATION SPECIFICATIONS

Syntax:

```

DEVICE <device name>
TYPE = <device type>
<device attributes>
END
    
```

Summary of Typical Attributes

Device Types	Required Attributes:	Default Units:
CPU	RATE	MIPS
DISK	RATE SEEK REVOLUTION_TIME	CHAR / MSEC MSEC MSEC
SERVER	RATE	OPS / MICROSEC
DEFINED	OPERATION TIME	___ MSEC
DEFINED_CPU	OPERATION TIME	___ MSEC

CONFIGURATION SPECIFICATION EXAMPLES

```

DEVICE EX_CPU
TYPE = CPU
RATE = 1 & MIPS
END

MODULE USE IT
EST EX_CPU USAGE = 10,000 INS
&OR
EST EX_CPU USAGE = 10 MSEC
END
    
```

VI - 3

Notes

19

System Model Workshop

CONFIGURATION SPECIFICATIONS EXAMPLES

```

GLOBAL . DECLARATION
  L = .00224;
  S = .00186;
.
.
.
DEVICE  EX2_CPU
  TYPE = DEFINED_CPU
  OPERATION ASSIGN_S
    TIME = L + S  MSEC
  OPERATION MULT_S
    TIME = L + M + LALS + S MSEC
    (etc.)
  END
.
.
MODULE  USE_2
  EST EX2_CPU USAGE = 860 ASSIGN_S
  END
  
```

L S W. 7

CONFIGURATION SPECIFICATION EXAMPLES

```

DEVICE  EX_DISK
  TYPE = DISK
  SEEK = 20           &MSEC
  REVOLUTION_TIME = 5 &MSEC
  RATE = 1000        &CHAR/MSEC
  END

Early life cycle specifications:
MODULE  USE_DISK
  EST  EX_DISK  USAGE = 25 MSEC
  END

Later - define files assigned to devices and read,
write operations
  
```

L S W. 8

VI - 4

[illegible]

System Model Workshop

CONFIGURATION SPECIFICATION EXAMPLES

```
DEVICE EX_SERVER
TYPE = SERVER
RATE = .000001 & OPS PER MICROSEC
& 1 FRAME / SEC

END

MODULE USE_SERVER
EST EX_SERVER USAGE = 1 OPS
&OR
EST EX_SERVER USAGE = 1000 MSEC

END
```

L S VI. 9

CONFIGURATION SPECIFICATION EXAMPLES

```
DEVICE EX_DEFINED_COM_LINE
TYPE = DEFINED
OPERATION = XFERS
TIME = 250 MSEC
& 12K block @ 9600 bits / sec
& Includes some overhead

END

MODULE EX_USE
EST EX_DEFINED_COM_LINE
USAGE = 2 XFERS

END
```

L S VI. 10

VI - 5

Notes

137

System Model Workshop

WORKLOAD SPECIFICATIONS

Syntax:

```

WORKLOAD <name>
  TYPE = <workload type>
  <workload attributes>
END
    
```

Summary of Typical Attributes

Workload Types:	Required Attributes:	Default Units:
INTERACTIVE	USERS THINK_TIME MPL JOB_STREAM	SECS JOBS (CLOSED)
TRANSACTION	ARRIVAL RATE JOB_STREAM MPL	JOBS/HR JOBS (OPEN)
PERIODIC	ARRIVAL_RATE JOB_STREAM MPL	JOBS/HR JOBS (Regular arrivals)
CYCLE	MPL JOB_STREAM	JOBS (Closed, Persistent Batch)

LS VI. 11

WORKLOAD SPECIFICATION EXAMPLES

```

WORKLOAD TIMING_LOOP
  TYPE = PERIODIC
  JOB_STREAM = HZ_20
  ARRIVAL_RATE = 20 * 3600 & 20 Frames per sec
END
    
```

```

WORKLOAD POOR_USER
  TYPE = INTERACTIVE
  USERS = 10
  THINK_TIME = 15 & SECS
  MPL = UNCONSTRAINED & NO MEMORY WAIT
  JOB_STREAM = DO_WORK
END
    
```

LS VI. 12

VI - 6

Notes

L's

System Model Workshop

WORKLOAD SPECIFICATION EXAMPLES

```

WORKLOAD   REQUESTS_FM_ADMIRAL
TYPE = TRANSACTION
ARRIVAL_RATE = 360      &1 REQUEST EVERY 10 SEC
MPL = UNCONSTRAINED
              & NO MEMORY WAIT
JOB_STREAM = URGENT_PROCESS
END
    
```

```

WORKLOAD   RECORD_FLIGHT_PLANS
TYPE = CYCLE
MPL = 1      & 1 AT A TIME
JOB_STREAM = UPDATE_DB
END
    
```

L S VI. 13

SPECIFYING SYSTEM EXECUTION MODEL PARAMS IN POD

Model Parameters	POD
<u>Workload Definition</u>	WORKLOAD.SPECIFICATION
One workload per scenario	
Workload Intensity	
Open models	TYPE = TRANSACTION
	ARRIVAL_RATE =
Closed models	TYPE = INTERACTIVE
	USERS =
	THINK_TIME =
<u>Execution Environment</u>	CONFIGURATION.
	SPECIFICATION
Devices	
Service rates	MIPS, MSEC, operations & time
<u>Software Execution</u>	MODULE.SPECIFICATION
Execution Structure	CASE, LOOP, CALL
<u>Resource Requirements</u>	
# visits	# operations or instructions
mean service time	service time at devices

L S VI. 14

VI - 7

Notes

LS

System Model Workshop

DYNAMIC ANALYSIS

1. Create SDF
2. Invoke POD
3. READ <sdf file name>
4. Check static analysis reports
5. GO
 - & Runs performance calculator
 - & Prints principal results report:
 - & Workloads, response time,
 - & throughput, % CPU utilization
6. Produce dynamic reports: *(menu driven)*
 - UT & Total utilization of each device and
 - & Utilization per workload
 - RP & Response time profile per workload
 - & and residence time at each device
 - QL & Queue length of each workload
 - & at each device
 - TP & Throughput and maximum throughput
 - & of each workload at each device
 - SV<device> & Server report gives utilization,
 - & queue length, response time,
 - & and wait statistics for each
 - & workload at the device
7. QUIT & Terminates execution

L S VI. 15

EVALUATION COMMANDS

SET Changes the values of parameters and attributes

SET <parameter> = <new value>

Examples:

SET TRACKS = 200

SET OBJECTS = 100

SET <name> <attribute> = <new value>

Examples:

SET EX_CPU RATE = 2 &MIPS

SET REQUESTS_FM_ADMIRAL ARRIVAL_RATE = 60

New value can be an expression

OLD $\frac{+}{-} \frac{<number>}{/}$

Example:

SET REQUESTS_FM_ADMIRAL

ARRIVAL_RATE = OLD * 2

L S VI. 16

VI - 8

Notes

LS

System Model Workshop

EVALUATION COMMANDS

? Prints the current value of parameters and attributes

Parameters

? <name>

Examples:

? PIXELS

? TRACKS

Attributes

? <name> <attribute>

Examples:

? REQUEST_FM_ADMIRAL_ARRIVAL_RATE

LIST

Reports current values for multiple components

LIST

CLASSES
DEVICES
WORKLOADS
PARAMETERS
MODEL
.
.
.

& Lists current model in PC format

SYSTEM MODEL WORKSHOP

EXERCISE 1

1. Edit the SDF named sample. It is a template that specifies most of the SDF data for the sample model in Slide V-13.
2. Specify the arrival rate, device service rates, and module resource requirements shown in the slide.
3. Solve the model. You have successfully solved this model when your results match the slide.

Notes

LS

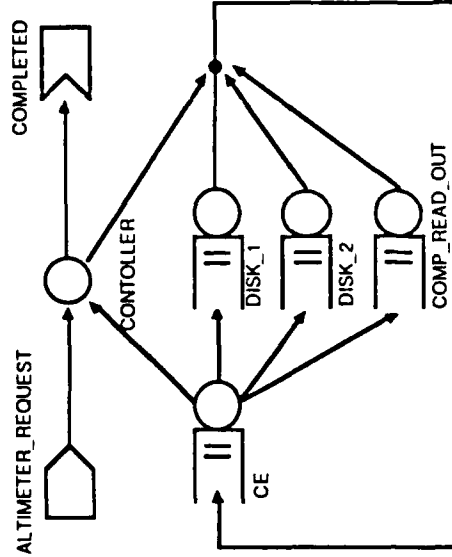
System Model Workshop

EXERCISE 2

1. Edit the SDF named faa_1. It is a template for the following model.
2. Use the information on the following ^{slides} to derive the model parameters for the SDF. Make the appropriate modifications to the SDF and solve the model. When you successfully solve the model, fill in the first column of results on the solution page.

L S VI 18

SYSTEM CONFIGURATION



Service rates

CE	.286 MIPS
DISK_1	100 MSEC
DISK_2	100 MSEC
COMP_READ_OUT	16.81 CHAR/MSEC
CONTROLLER	10 SECS

L S VI 20

VI - 10

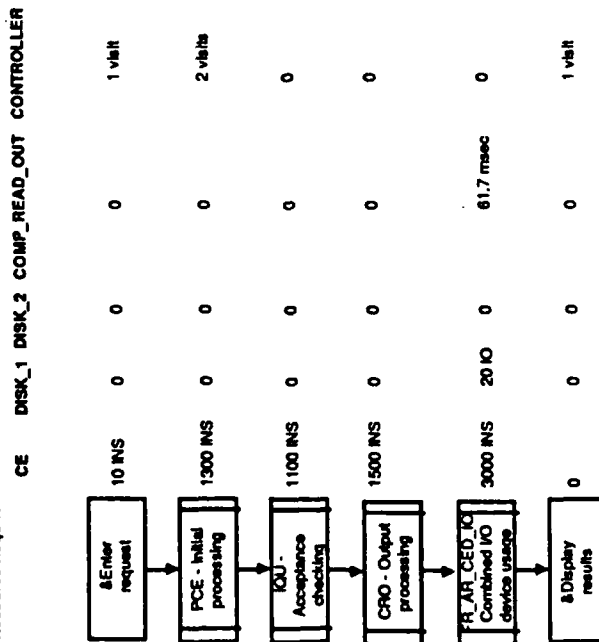
A full page of graph paper with a uniform grid of small squares. The grid covers most of the page, leaving margins at the top, bottom, and sides. In the top right corner, there is a small rectangular box containing the number "19".

5

System Model Workshop

SOFTWARE EXECUTION

Scenario: Altimeter_requests
1800 requests/hr. word case
Resource Requirements



L S VI - 21

STUDIES

1. Modify the model and change the service time for DISK_1 to 50 msec. Solve the revised model and enter the results in column 2.
2. Modify the model and set the service time for both DISKS to 100 msec. Equally balance the I/O requests in FR_AR_CED_IO between the two disks. Enter the results in column 3.
3. Suppose we modify the software design and thus change the resource requirements to the following:

Component	CE	DISK_1	DISK_2	COMP	READ_OUT	CONTROLLER
& Enter request	10	0	0	0	0	1
PCE	10950	0	0	0	0	0
IOU	10170	0	0	0	0	0
CRO	13802	0	0	0	0	0
FR_AR_CED_IO	3000	1	1	61.7	0	0
&Display	0	0	0	0	0	1

Revise the model, solve it, and enter the results in column 4.

4. Change the arrival rate to the average case, 88 requests per hr., reevaluate, and enter the results in column 5.

L S VI - 22

VI - 11

	EXERCISE 2: RESULTS				
STATISTICS	CASE 1:	CASE 2:	CASE 3:	CASE 4:	CASE 5:
	Original	Faster	Balanced	Reduce	Average
	specs	disk	load	work	arrival rate
-----	-----	-----	-----	-----	-----
Response time					
Utilizations:					
CE					
DEV1					
DEV2					
COMP_READ_OUT					
CONTROLLER					
Residence time:					
CE					
DEV1					
DEV2					
COMP_READ_OUT					
Queue length:					
CE					
DEV1					
DEV2					
COMP_READ_OUT					

Notes

LS

System Model Workshop

EXERCISE 3 Multiple Workloads

1. Edit the file *tas_bigger* and make the following modifications:

- Change the CE processing rate to 1 MIPS.
- Add another workload with the following software execution structure:

Scenario: OTHER_MISC_PROCESSING

Arrival rate = 3600 & REQUESTS / HR

Component	CE	DISK_1	DISK_2	COMP_READ_OUT	CONTROLLER
&Gel request	110	0	0	0	1
PCE	10950	0	0	0	0
IOU	10170	0	0	0	0
CRO	13802	0	0	0	0
CSF	41847	0	0	0	0
&MISC PROC	3000	8	6	25.5	1

2. Solve the model and enter the results in the table on the following page.

EXERCISE 4 (optional)

Run the performance calculator on the signal processing SDF. What is the system throughput (frames/sec.)? Modify the assignment of modules to processors to improve throughput. Re-solve and quantify the performance improvement.

		EXERCISE 3: RESULTS	
STATISTICS	Altimeter Requests	Other Misc. Processing	All
-----	-----	-----	-----
Response time			
Utilizations:			
CE			
DEV1			
DEV2			
COMP_READ_OUT			
CONTROLLER			
Residence time:			
CE			
DEV1			
DEV2			
COMP_READ_OUT			
Queue length:			
CE			
DEV1			
DEV2			
COMP_READ_OUT			

SPE Methodology

L
S

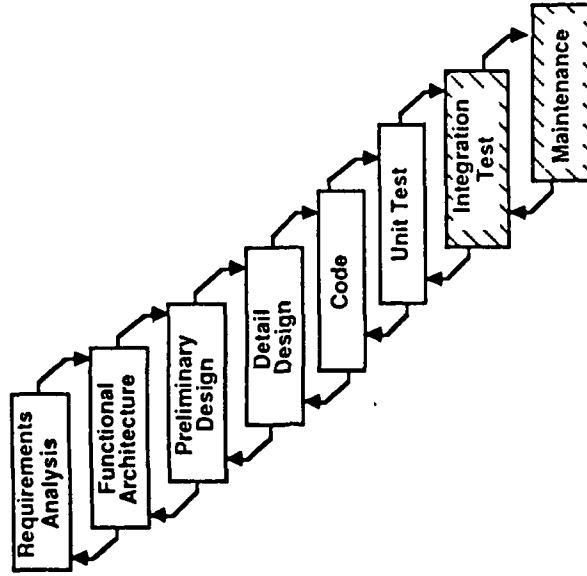
SPE Methodology

OBJECTIVES

- > To introduce software development models
- > To show the relationship of SPE
- > To give an overview of SPE objectives and activities

L S VII-1

WATERFALL MODEL

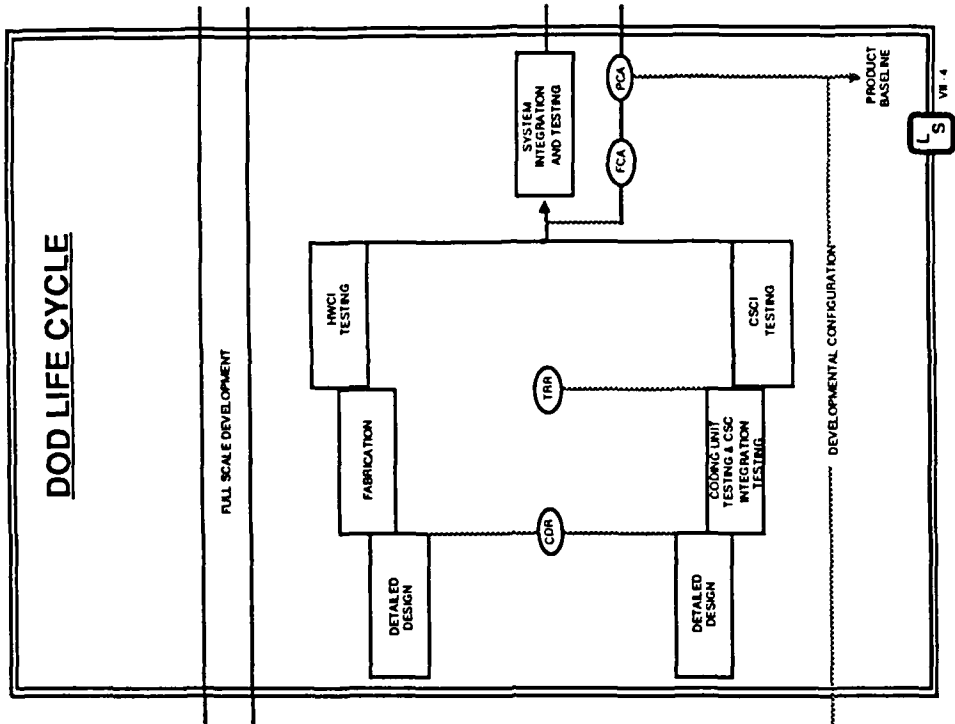
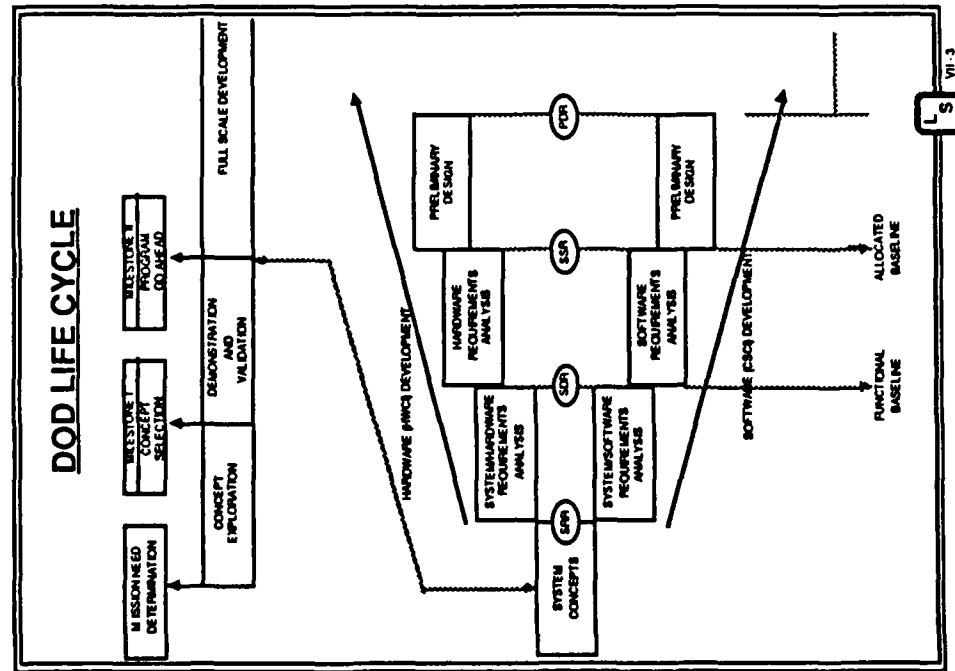


VII - 1

[illegible]

5

SPE Methodology



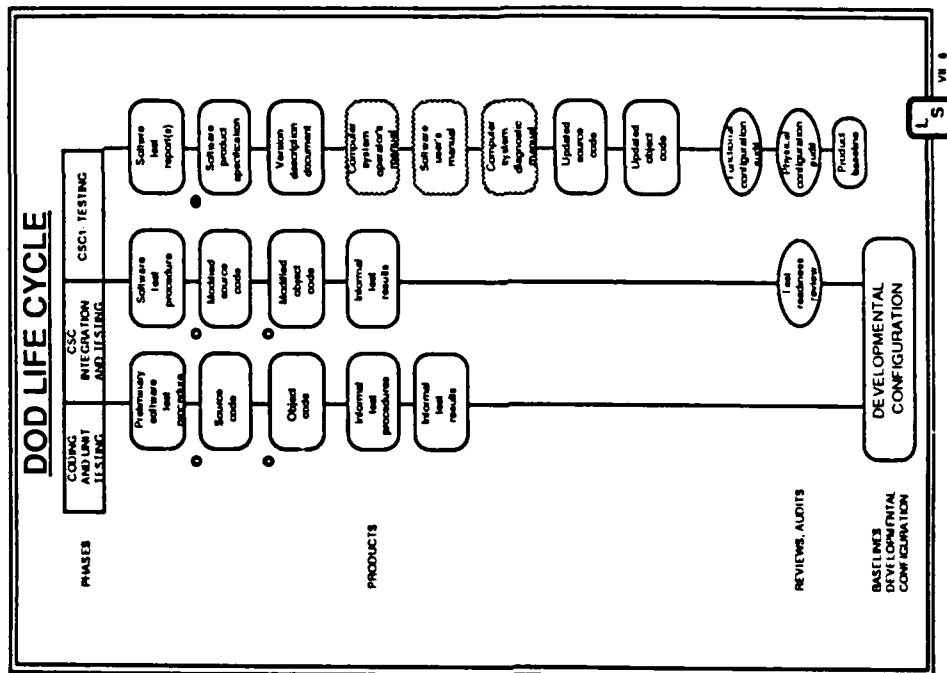
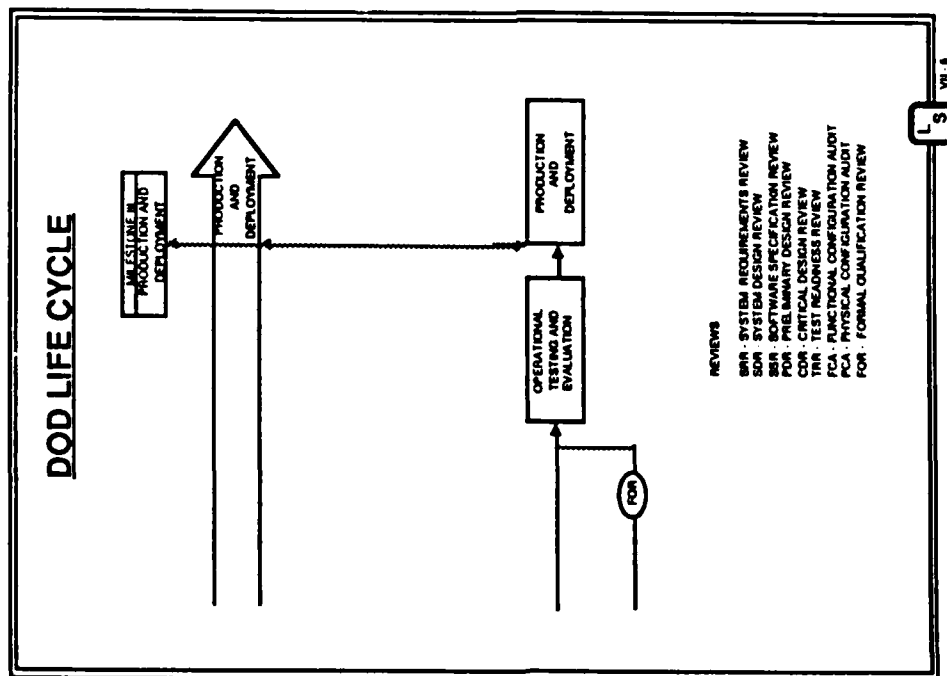
VII - 2

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

15

SPE Methodology

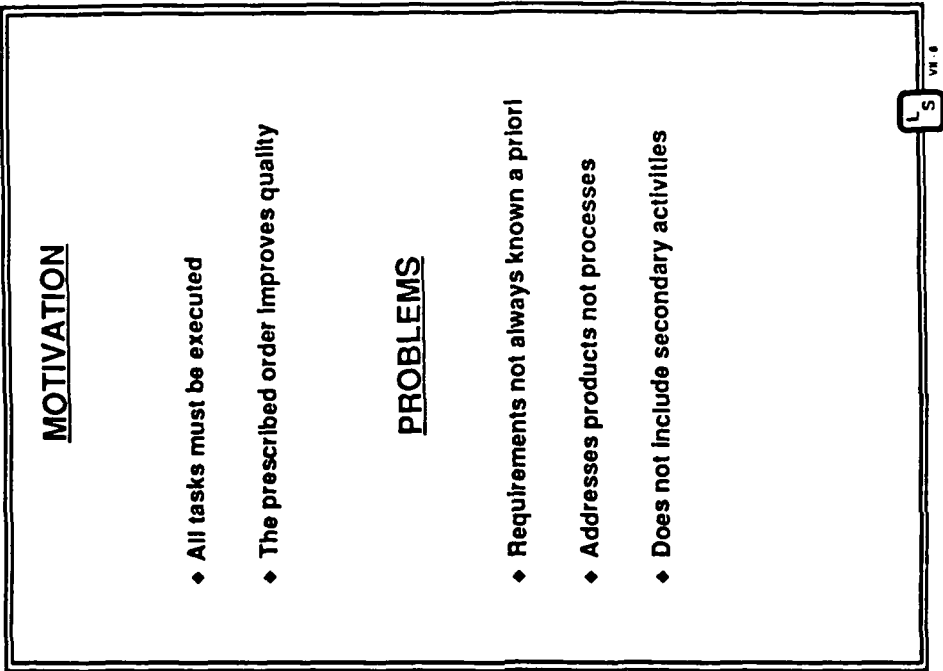
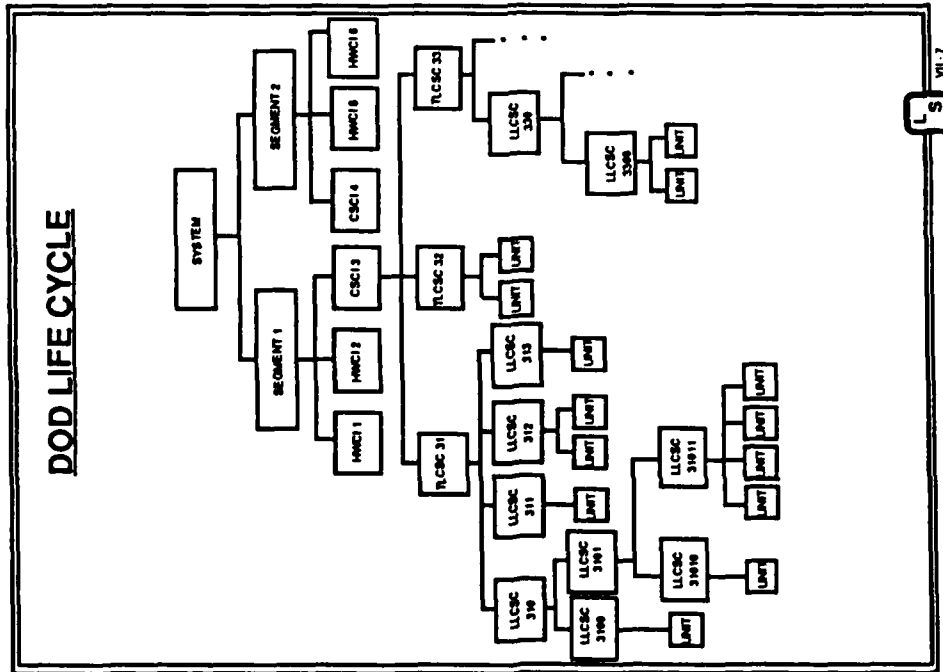


VII - 3

[illegible]

SL

SPE Methodology



VII - 4

Notes

LS

SPE Methodology

INCREMENTAL BUILD

" A phased implementation of system subsets "

- ♦ 'System' runs earlier
- ♦ Early resolution of problems
- ♦ Advanced planning more difficult
- ♦ Prioritize enhancements

How big is an increment?

L S VII 9

PROTOTYPING

" Build a replica as an experiment before building the actual system "

- ♦ Refine the requirements
- ♦ Demonstrate feasibility
- ♦ Learn from experience

Problems?

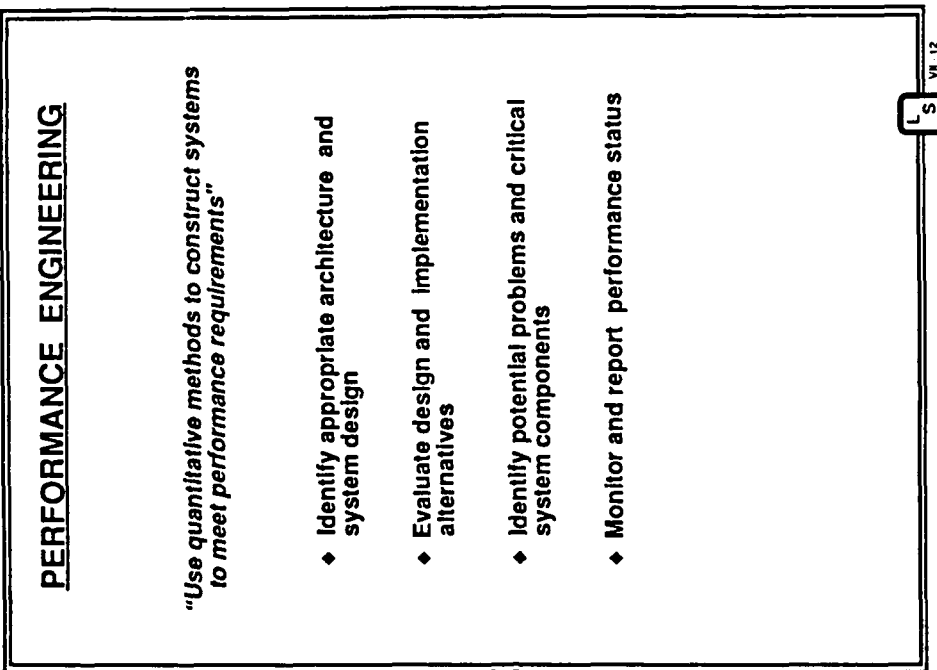
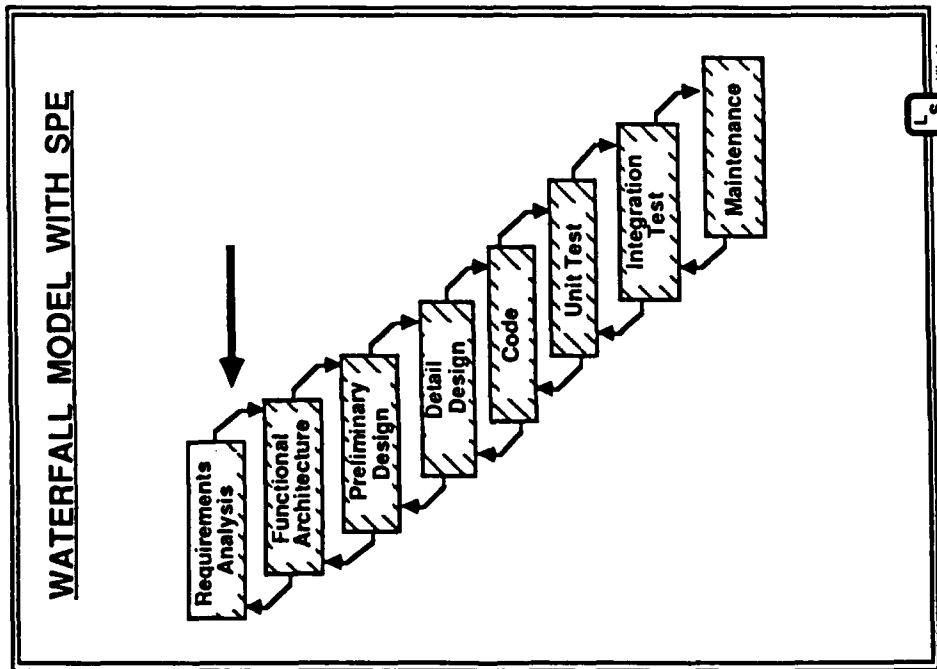
L S VII 10

VII - 5

Notes

LS

SPE Methodology



VII - 6

Notes



SPE Methodology

METHODOLOGY OVERVIEW

FOR each development stage DO

BEGIN

REPEAT

Develop design concept

Model it

Evaluate model solution

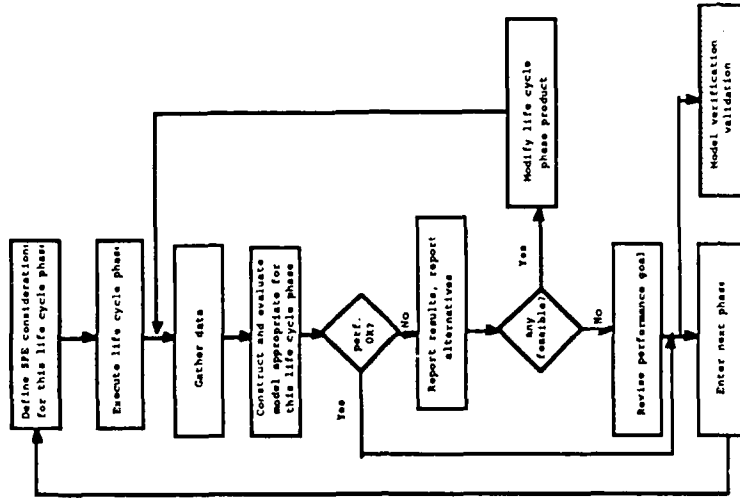
UNTIL predicted behavior is satisfactory

Proceed with design synthesis

END

L S VII-13

SPE METHODOLOGY



L S VII-14

VII - 7

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

L S

SPE Methodology

SPE METHODS INCLUDE

- ♦ Performance Walkthroughs
- ♦ Data Gathering Strategies
- ♦ Compensation for Uncertainties
- ♦ Data Presentation & Tracking
- ♦ Model Verification & Validation
- ♦ General Principles for Performance-oriented Design

L S
VI-18

SPE CONSIDERATIONS

- ♦ Precise performance goals
Response time
Throughput
Resource budgets
- ♦ Benchmark scenarios
- ♦ Workload Intensity
- ♦ Performance questions

L S
VI-18

VII - 8

Notes

LS

SPE Methodology

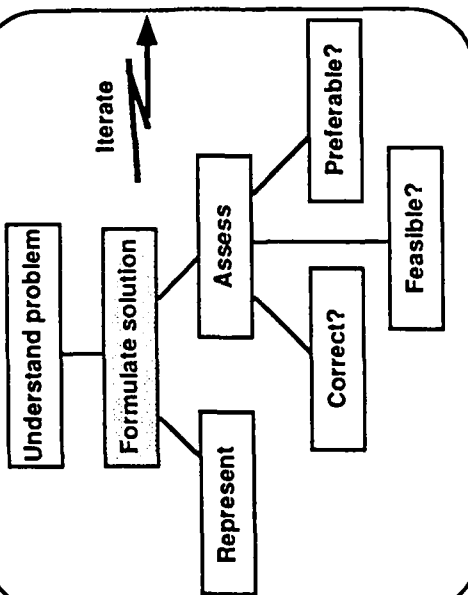
SAMPLE LIFE CYCLE QUESTIONS

Life Cycle Stage	Performance Considerations
Requirements Analysis & Functional Architecture	What are the key operational scenarios? How can we meet functional and performance requirements? Approximately how much of the available computer power will be used?
Preliminary Design	Is the expected performance of this design satisfactory? Which parts of the system should be watched closely?
Detailed Design	Have changes occurred which affect earlier predictions? What is a more realistic estimate of the projected performance?
Implementation & Integration testing	How does the performance of the implementation alternatives compare? Have any unforeseen problems arisen? What are the resource requirements of the critical components?
Maintenance & Operation	What is the effect of the proposed modifications? Which modification alternative is "best"? What are the long-range configuration requirements to support future use?

L S VII-17

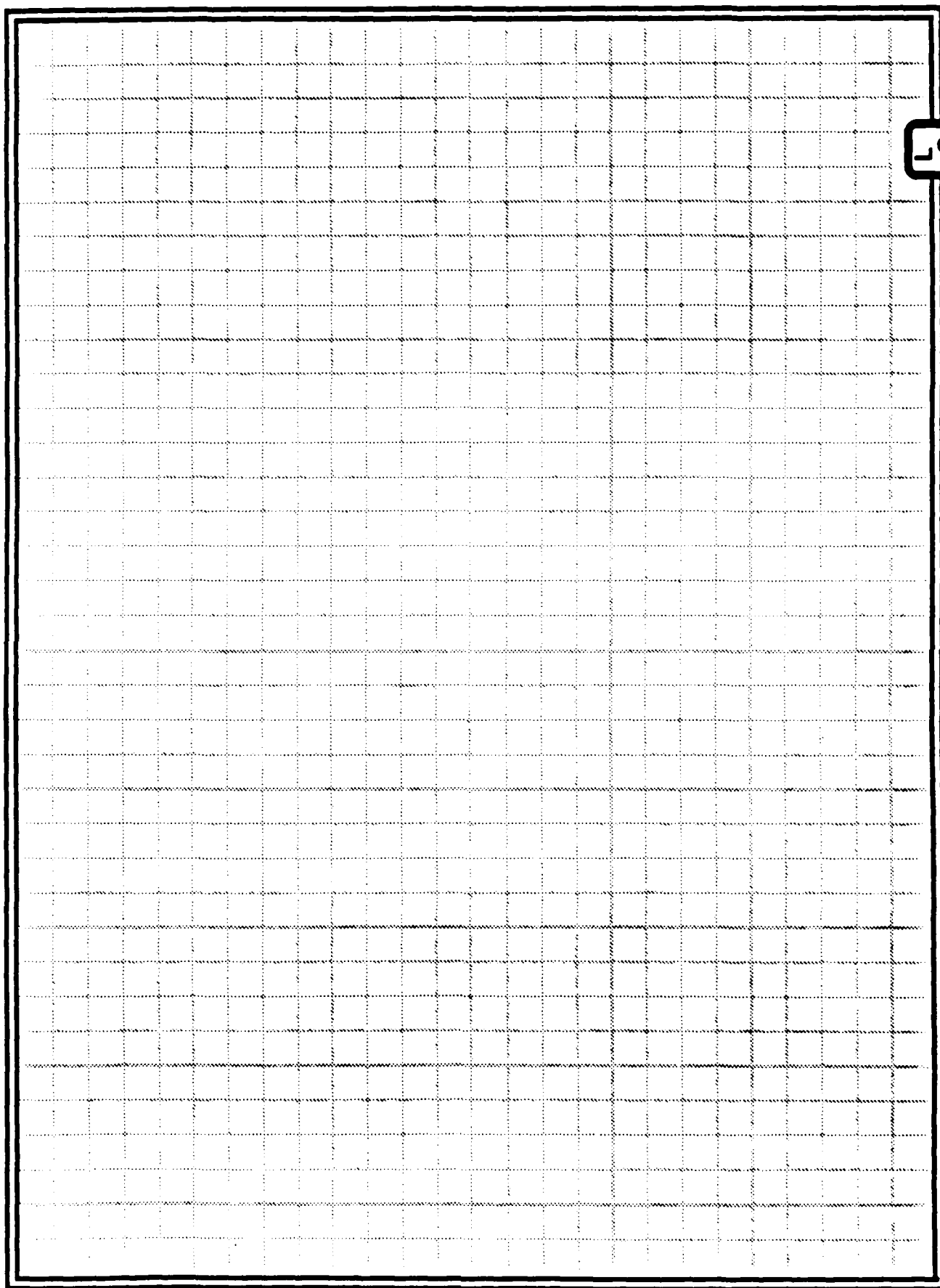
PERFORMANCE-ORIENTED DESIGN PRINCIPLES

Engineering Design Process



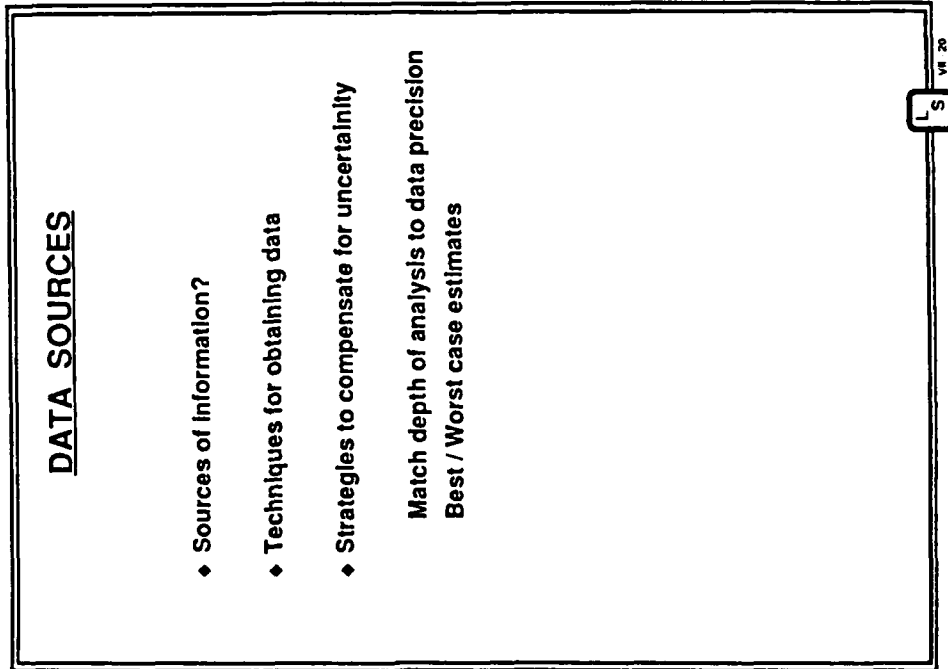
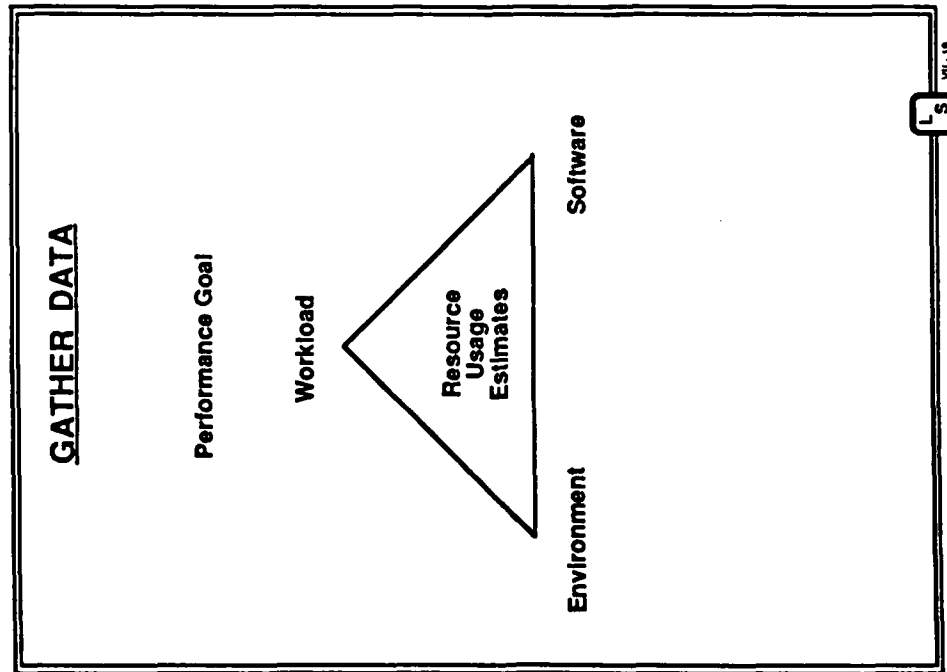
L S VII-18

Notes



L S

SPE Methodology

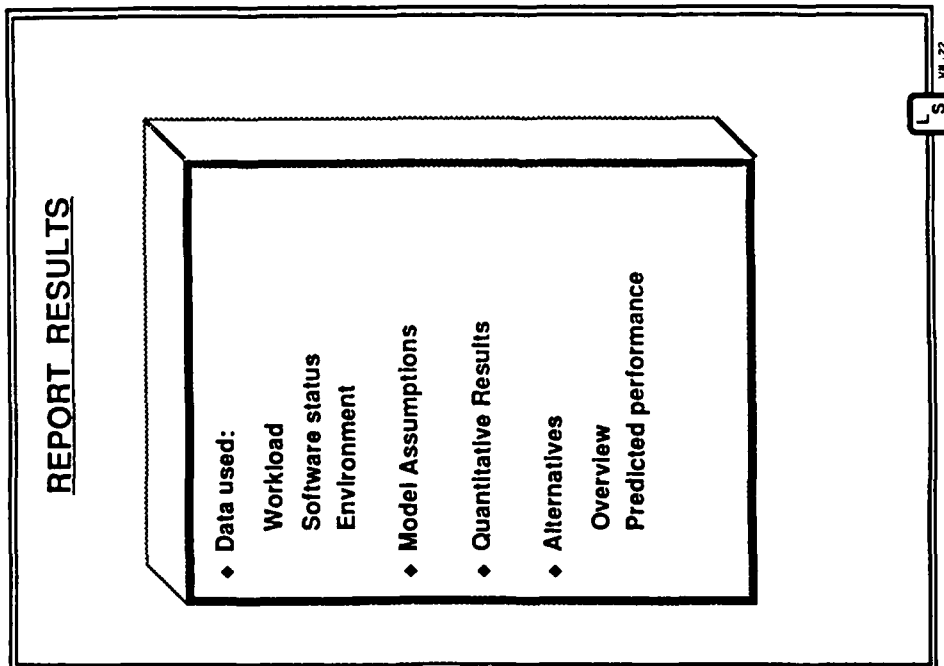
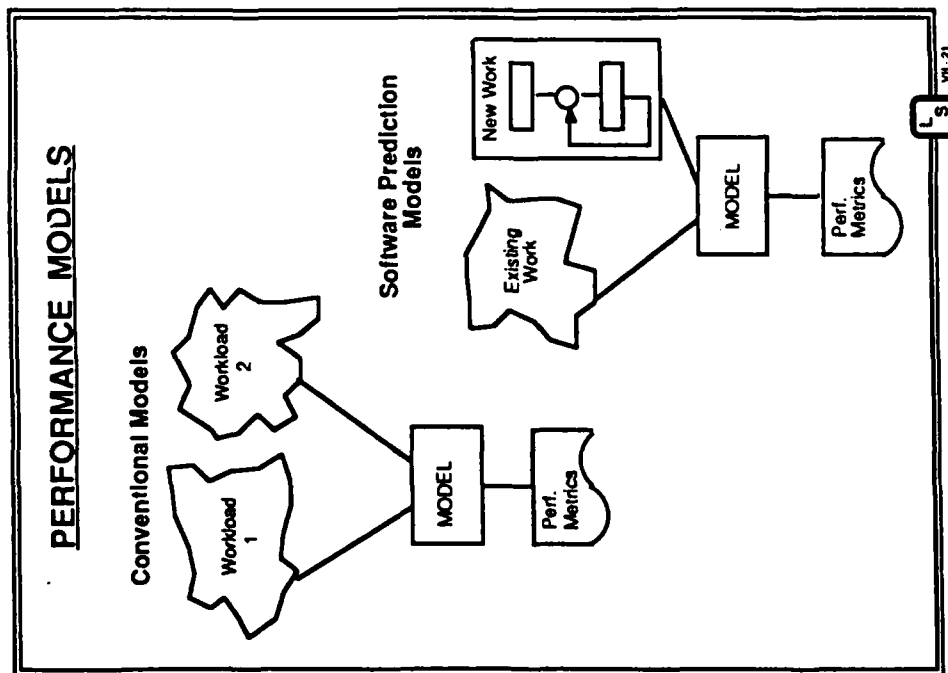


VII - 10

Notes

15

SPE Methodology

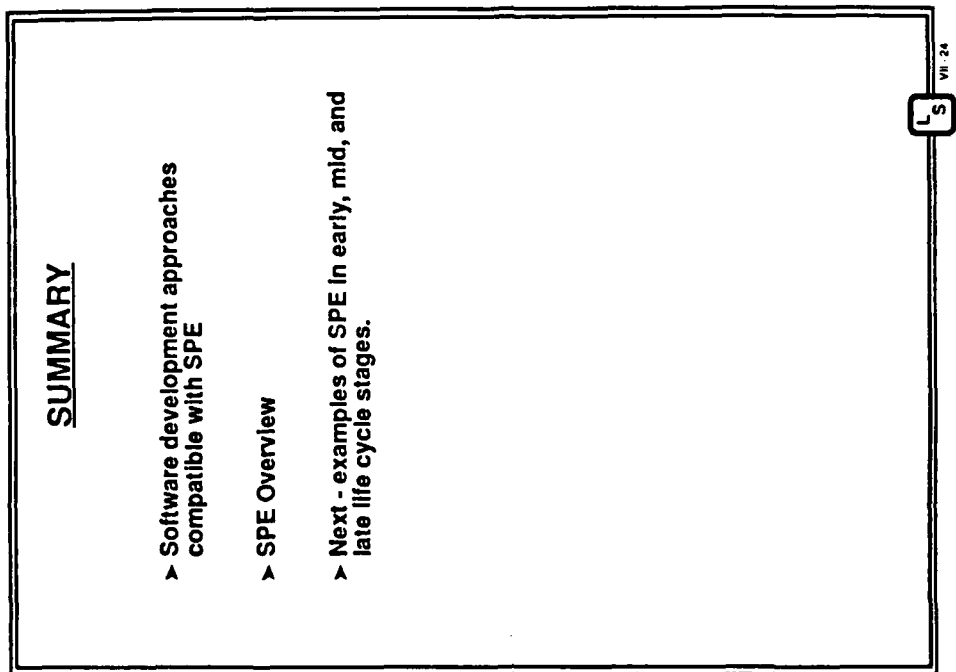
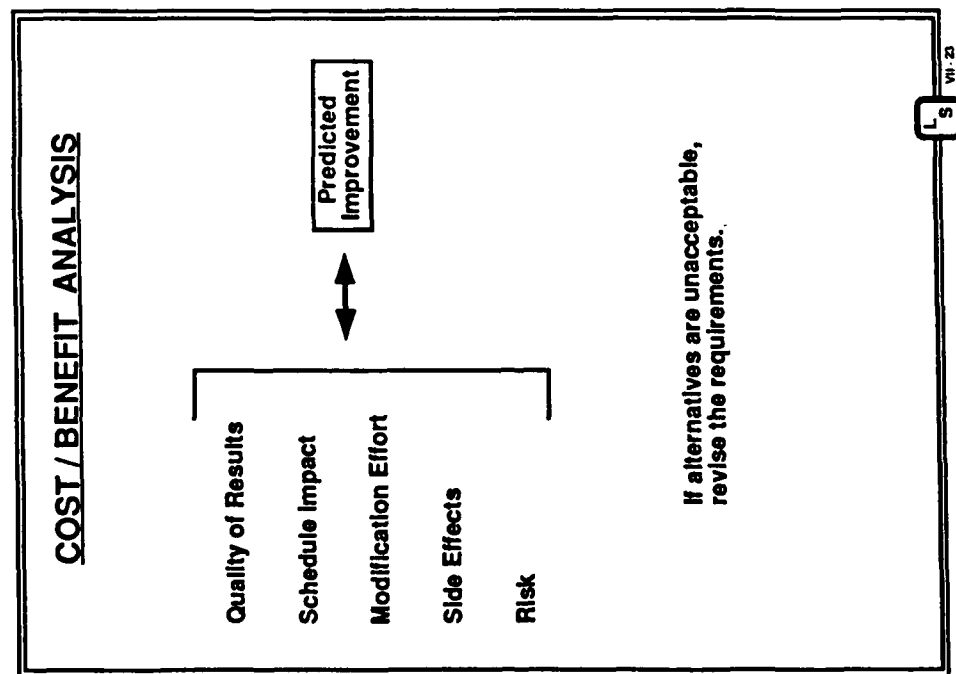


VII - 11

Notes

LS

SPE Methodology



Life Cycle Examples

L_S

Life Cycle Examples

OBJECTIVES

- > Demonstrate SPE techniques and models for actual case studies
- > Demonstrate differences in life cycle stages
- > Demonstrate tactics for understanding problems gathering data, creating & evaluating models

L S VII-1

PART I

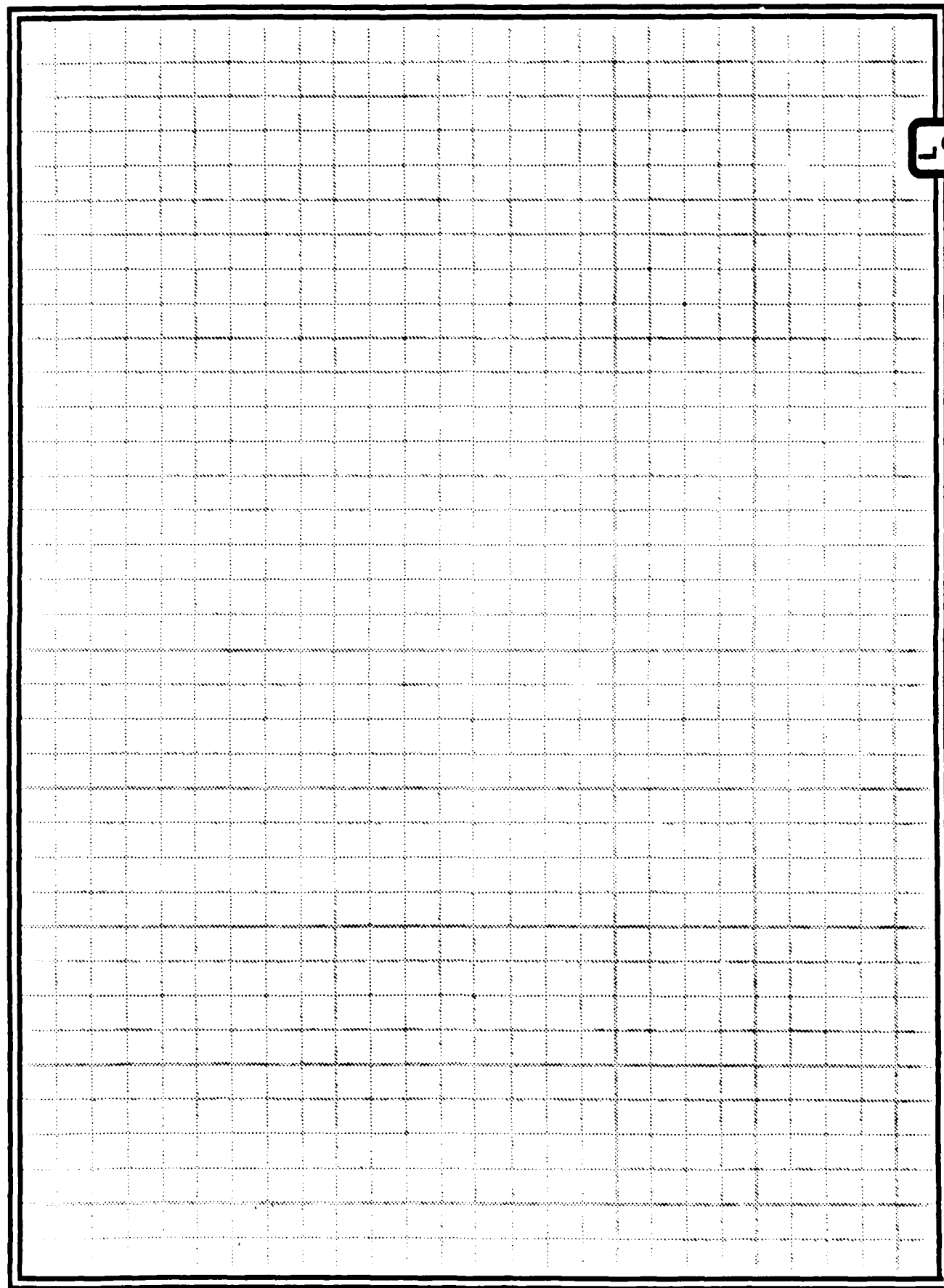
Early Life Cycle Studies

L S VII-2

VIII - 1

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes



15

Life Cycle Examples

CASE STUDY BACKGROUND

- ♦ FAA upgrade for En Route Air Traffic Control (ATC) computer system
 - Close to capacity limits
 - Traffic growth expected
 - Maintenance & support of aging systems problematic
 - Automate more functions
- ♦ Alternatives
 - Replacement architecture compatible with existing software
 - Distributed network architecture
 - Hybrid - compatible central processor & incremental addition of distributed functions & hardware

L S VII 3

SYSTEM FUNCTIONS

- ♦ ATC levels: National, en route, airport, aircraft
- ♦ En route centers: Responsible for control regions comprised of 3D airspace of sectors
- ♦ Primary functions
 - Radar data processing (RDP)
 - Process target data from surveillance sites
 - Initiate tracking on target data
 - Associate tracks with flight plans
 - Display aircraft track & info to controllers
 - Flight data processing (FDP)
 - Accept flight plan data
 - Update as aircraft passes through airspace
 - Controllers
 - Generate commands and clearances to aircraft
 - Assure conflict-free, expeditious travel along flight plan
- ♦ Key requirements
 - Information reliability
 - Process input messages within specified time

L S VII 4

VIII - 2

Notes

15

Life Cycle Examples

PERFORMANCE PROJECT

Create & validate operational model
Study alternatives
Demonstrate viability of models

Steps

1. Formulate modeling approach:
Characterize message input → response
Aggregate into message classes
2. Identify key message types, their input & output
3. Determine application program elements activated by each message type
4. Determine resource requirements of program elements
5. Quantify resource requirements of "other work"

L S VII 4

WORKLOAD CHARACTERIZATION

100 message types → 5 classes

1. Flight plan readout (FR) from computer entry device (CED)
2. Flight plan readout (FR) from Flight data entry & printout dev (FDEP)
3. Flight plan readout (FR) from Input output typewriter (IOT)
4. Altimeter request (AR) from CED
5. Other processing

Program elements and resource requirements:

Message	PE	Total CE (msec)
AR_CED	PCE	3025
	IQU	1529
	CRO	2992
FR_IOT	PIT	316
	IQU	1529
	CSD	265
	CSR	33

L S VII 4

VIII - 3

Notes

15

Life Cycle Examples

RESULTS

Validation

CE Utilization	FR_CED Response Time
POD 71.5	.35
Measurements 73	.33
In general ±10%	±30%

Once models are formulated, other studies are easy
Faster hardware
Workload volume increase
Sensitivity to derived resource requirements

Common problem -- getting accurate measurements with desired granularity
Wanted:
By message type -- PE's called, resource requirements
Measured:
Total # times PE's called
Total resource requirements
Also want end-to-end response times

Slow speed devices >> software processing time

LS

VII 7

DISTRIBUTED ATC STUDY

Workload components

1. En route surveillance target processing
2. Flight plan data
3. Controller operations - 2 controllers per sector
 - R: Radar - monitors spatial relationships
 - F: Flight plan data - assists R & enters & displays flight data
- (Estimate number of operations / minute / controller / sector)
4. Flight plan monitoring
5. Flow control processing
6. Weather reports
7. Predicted conflicts
8. Minimum safe altitude warnings
9. Data recovery

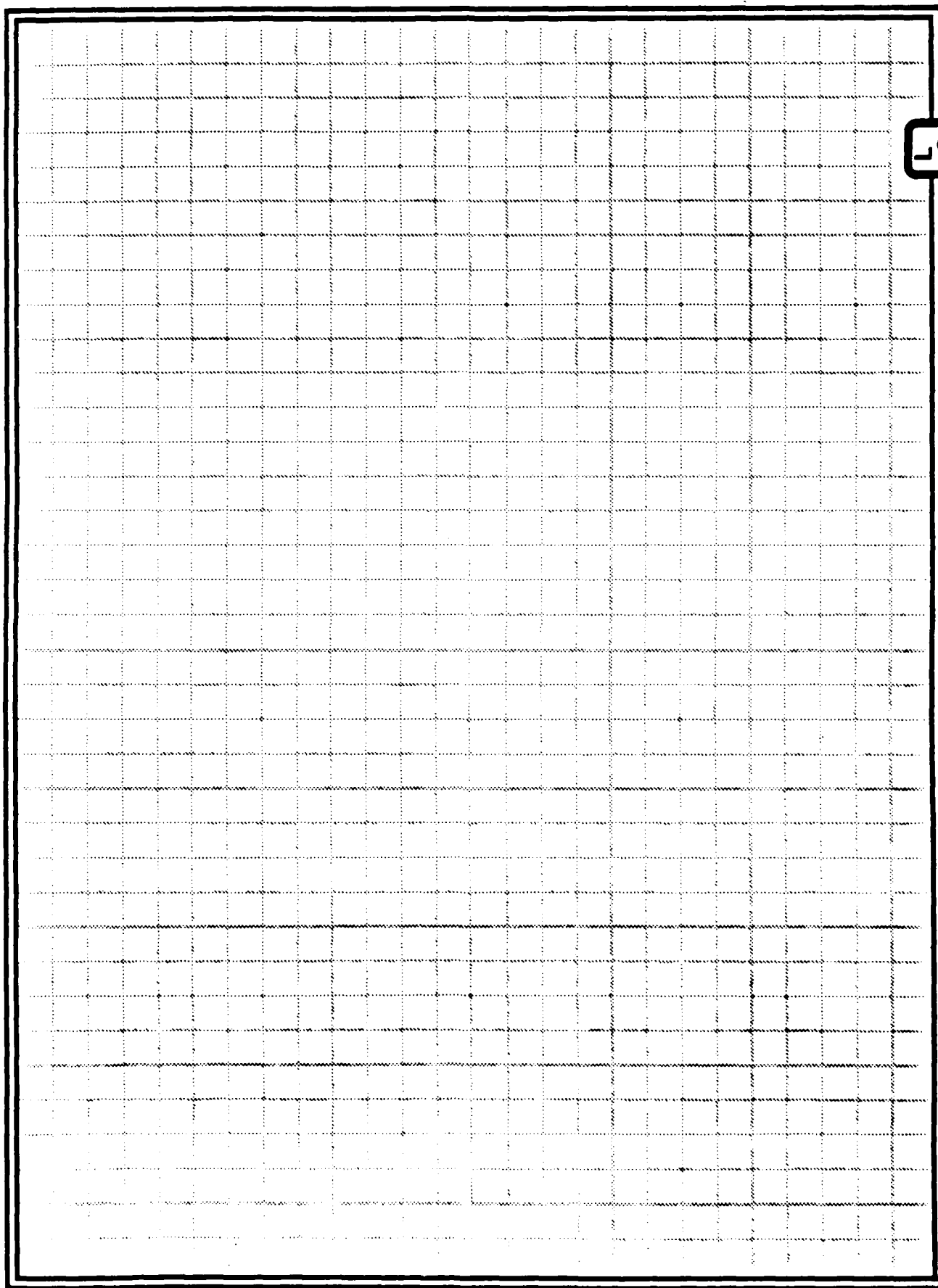
Estimate workload intensity based on historical data

LS

VII 8

VIII - 4

Notes



Life Cycle Examples

SOFTWARE PROCESSING

Walkthrough software components executed for each workload element

Example Surveillance data processing

1. Determine appropriate T sector processor
2. Build track list
3. Match target with flight plan data
4. Get target / track data for free tracking, target / track / flight data for flat tracking
5. Generate display

$$\begin{aligned} \text{Arrivals / rate} &= ((\text{departure_arrivals} + \\ &\text{overflights}) * \text{radar_scan}) / \text{no_sectors} \\ &= 3680 \text{ targets / sector / hour} \end{aligned}$$

L S VII 9

MODULE SPECIFICATIONS

Example MODULE FREE_FLAT_TRACKING

```
TEST FREE_FLAT_TRACKS
CASE 'FREE'      &57%
EST CPU_T USAGE = 80 OPS
CALL ICS_TRANSMISSION
CALL UPD_TRACK_DB
CASE 'FLAT'      &43%
EST CPU_T USAGE = 80 OPS
CALL ICS_TRANSMISSION
CALL UPD_TRACK_DB
ENDTEST
```

L S VII 10

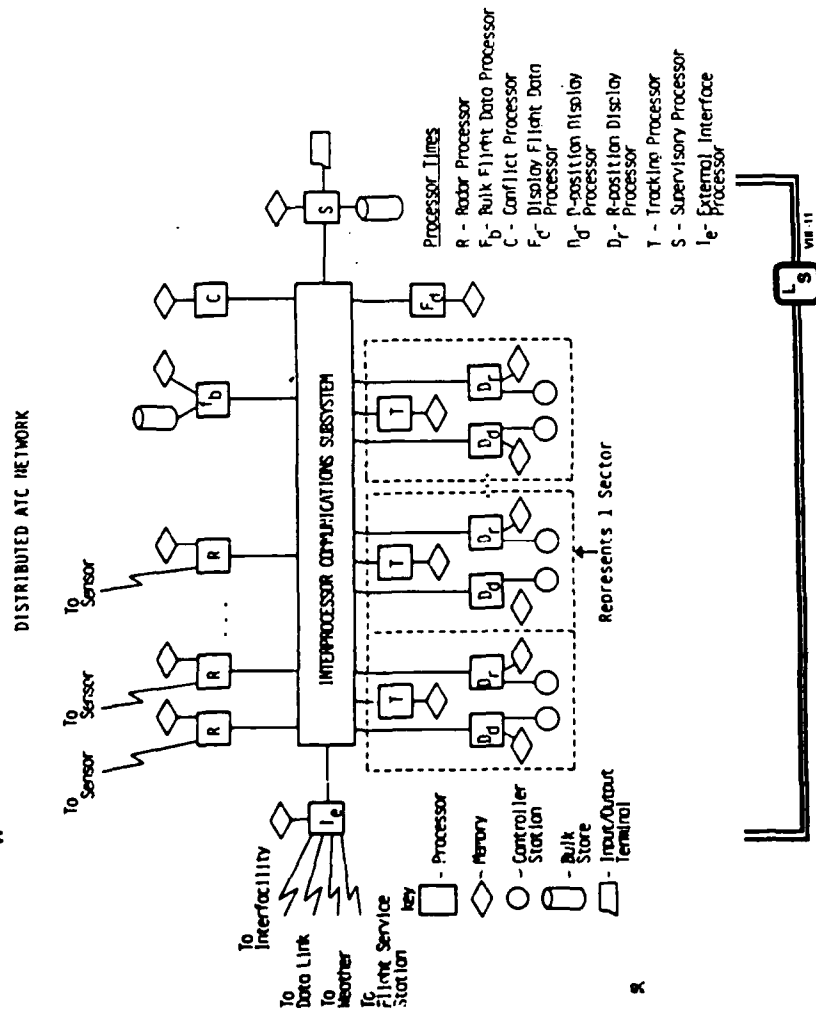
VIII - 5

Notes

A large rectangular area with a grid of dotted lines for writing notes. The grid is composed of horizontal and vertical dotted lines forming a series of small squares. The area is enclosed by a solid black border.

Life Cycle Examples

CONFIGURATION SPECIFICATIONS



STUDIES

- ◆ Best case models - evaluate feasibility of distributed processing
- ◆ Configuration requirements
 - Processor
 - Bus
 - Number, speed
- ◆ Critical areas

Case study results
Modeling convenience
FAA evaluations

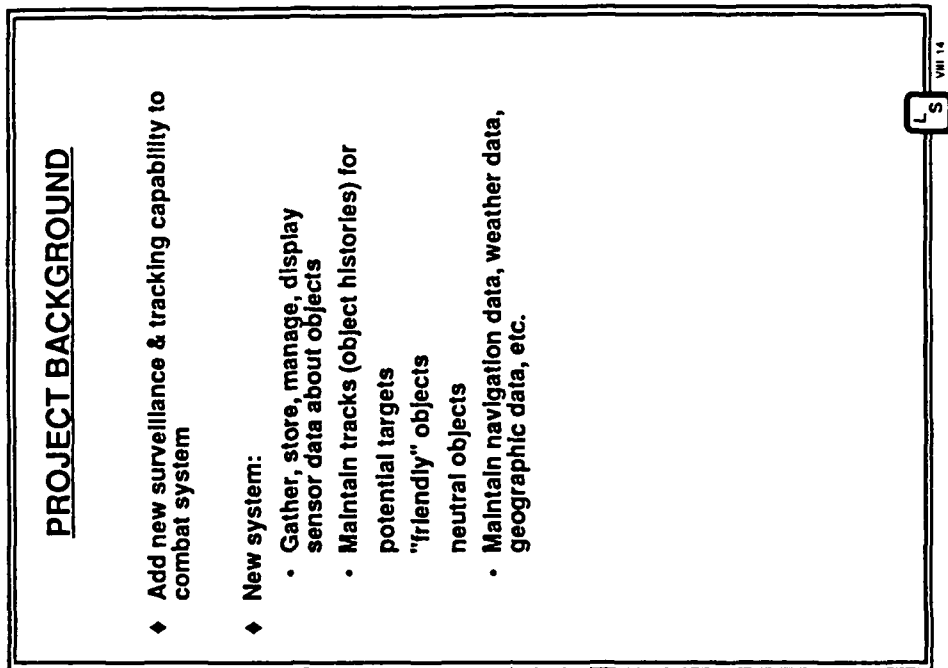
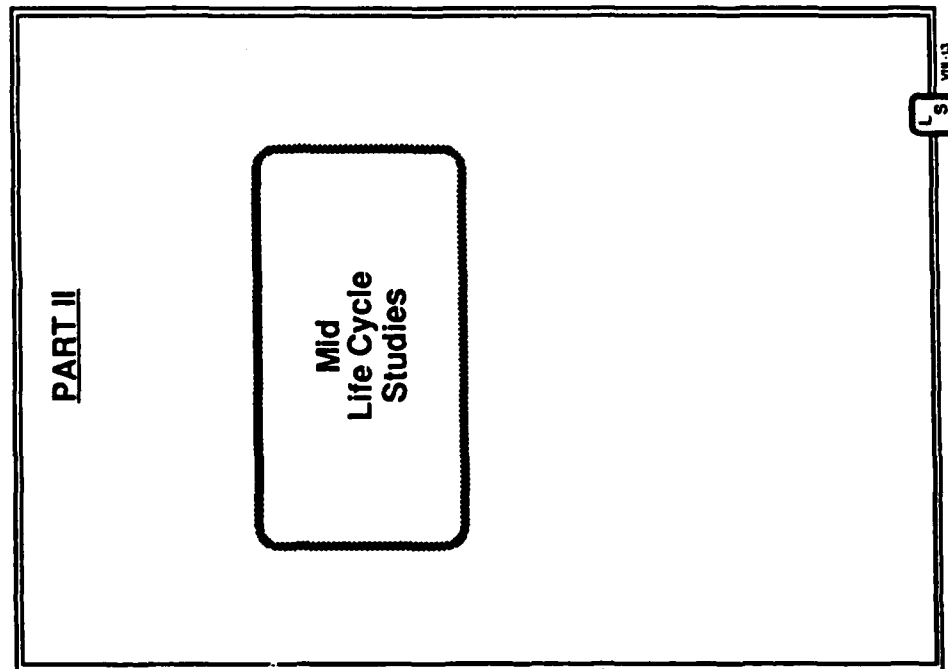
VIII 12

VIII - 6

Notes

15

Life Cycle Examples



VIII - 7

This image shows a full page of blank graph paper. The grid consists of small squares formed by thin, light gray lines. A thicker, darker gray border runs along the edges of the page. In the top right corner, there is a small black rectangular tab or piece of tape partially visible. The overall appearance is that of a clean, unused sheet of paper from a notebook or a set of drafting tools.

5

Life Cycle Examples

PERFORMANCE QUESTIONS

Early

- CPU sizing
- Expected response times
- Sensitivity to I/O speeds
- Effect of shared device contention

Mkd

- Table lookup vs direct computation algorithm
- Search strategy? linear, binary, compute index
- Partially implemented system match early model specifications?
- Meet performance objectives?
- Memory buffers vs. cached I/O device

Create models early
& add details as software evolves

LS

VIII-16

SYSTEM FUNCTIONS

Monitor Initialize Track file mgt. Sensor data mgt. Validate & correlate data Display control Track summary Tracks to combat control Track for quality check Tracks to communicate General display	Tactical Data Link-II Data transfer Display link data
Operator Training Data Links Commands	Operational Data Link Remote platforms Communication sites
Operating System Realtime control Centralized services	ComLat Control System Fire control etc.

Scenarios?

LS

VIII-16

Notes

5

Life Cycle Examples

WORKLOAD

- ♦ Operator Requests
- ♦ Inbound Tactical Data
- ♦ Inbound Operational Data

OS implicitly modeled by calls within module descriptions
Other options?

Choice: How to model operator requests
Specific Scenarios for key requests
Generic scenarios

L S VII-17

GENERIC MODEL OPERATOR REQUESTS

MODULE MONITOR
CALL INITIALIZE
: :
: : & MONITOR DISPLAY FUNCTIONS

TEST REQUEST_TYPE
CASE TRACK_SUMMARY... &20%
CASE TRACKS_TO_COMBAT... &10%
CASE TRACK_HIST_QUAL_CHK... &15%
CASE TRACK_SELECT... &15%
CASE GENERAL_DISPLAY &20%
: :
: :

ENDTEST

END

- ♦ Average behavior
- ♦ Worst case path
- ♦ Vary percent & study sensitivity
vs. key scenarios with specific performance targets
vs. combination

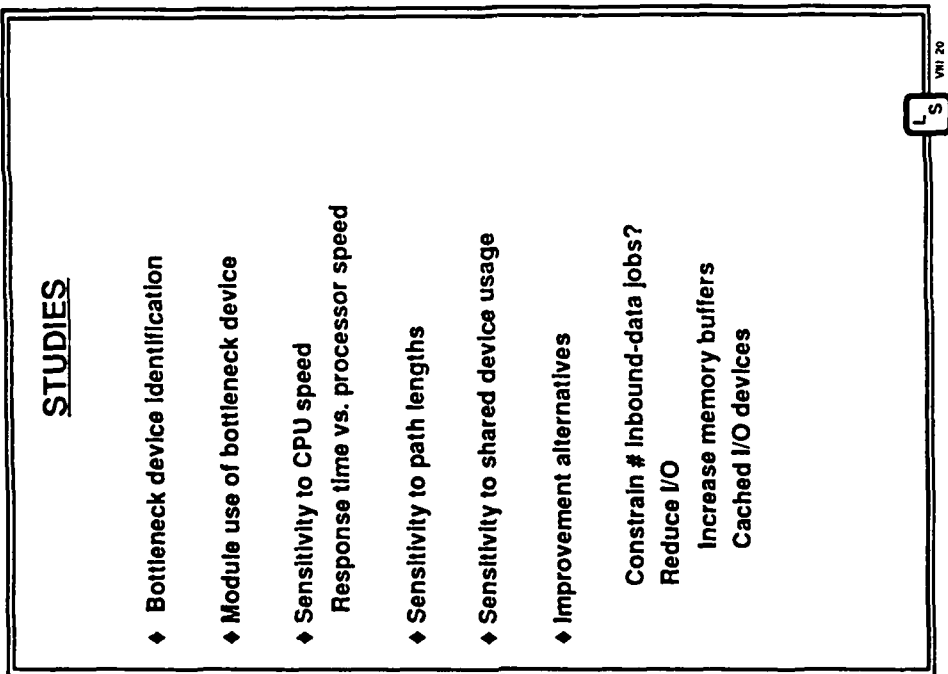
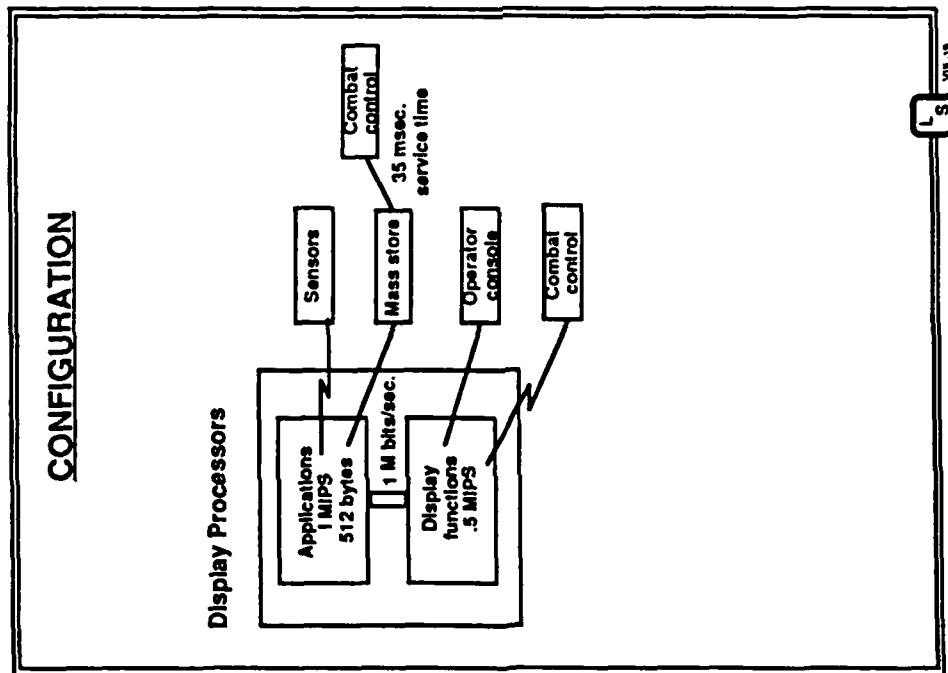
L S VII-18

VIII - 9

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

[illegible]

Life Cycle Examples



VIII - 10

Notes

A large rectangular grid for taking notes, with a dashed border and a small tab in the top right corner. The grid is composed of small squares, with a larger square in the center. The tab in the top right corner contains the letter 'S'.

Life Cycle Examples

SHARED DEVICE MODEL

- ◆ Contention delays at mass store device
- ◆ No increase in processor usage
- ◆ How to model?

CONSTRAIN # JOBS

- ◆ POD documentation refers to "memory" queue
-- actually a wait outside the system
- ◆ Set MPL for workload to constrained #

L S
VR 21

MASS STORE OPTIONS

- ◆ Increase memory buffers:
Use prameter to tweak "buffer misses" to govern # logical to physical I/O's
Vary BUF_MISS from 0 - 1 +
Study sensitivity

EXAMPLE:
MODULE SAMPLE
EST MASS USAGE = 2 * BMISS READ
END

- ◆ Cached I/O devices
Define different operations for device
Use case to regulate % of time reads found in cache

EXAMPLE:
DEVICE MASS TYPE = DEFINED
OPERATION READ
TIME = R_SEEK + R_ROTATE/2 + R_REC SIZE * R_XFER
MSEC
OPERATION CACHE_READ
TIME = R_REC SIZE * R_XFER MSEC
END

L S
VR 22

VIII - 11

[illegible]

Life Cycle Examples

SAMPLE RESULTS

Workload	Response	% CPU	% MASS	% DISP	% sensor link	% combat link
OP_REQ	6.7	15	9	.1	1.1	.1
TAC_DATA	6.9	6	35	0	0	0
OPER_DATA	6.9	6	35	0	0	0
Contraln . number of jobs						
		Response Time		Residency Time		
OP_REQ		7.2		4.2		
TAC_DATA		8.6		2.7		
OPER_DATA		8.6		2.7		
Slower CPU						
OP_REQ		23.8		7.0		
TAC_DATA		20.8		3.4		
OPER_DATA		20.6		3.4		

L S VII 23

OTHER STUDIES

- ♦ Table lookup vs. direct computation algorithm?
- ♦ Search strategy - linear, binary, compute index?
- ♦ Partially implemented system match specs?
- ♦ Meet performance objectives?
- ♦ Critical components

COMMON MID-LIFE CYCLE PROBLEMS

- ♦ Locally optimal algorithms that cause global problems
- ♦ Excess memory usage
- ♦ Synchronization problems
- ♦ Priority conflicts

Others?

How to detect?

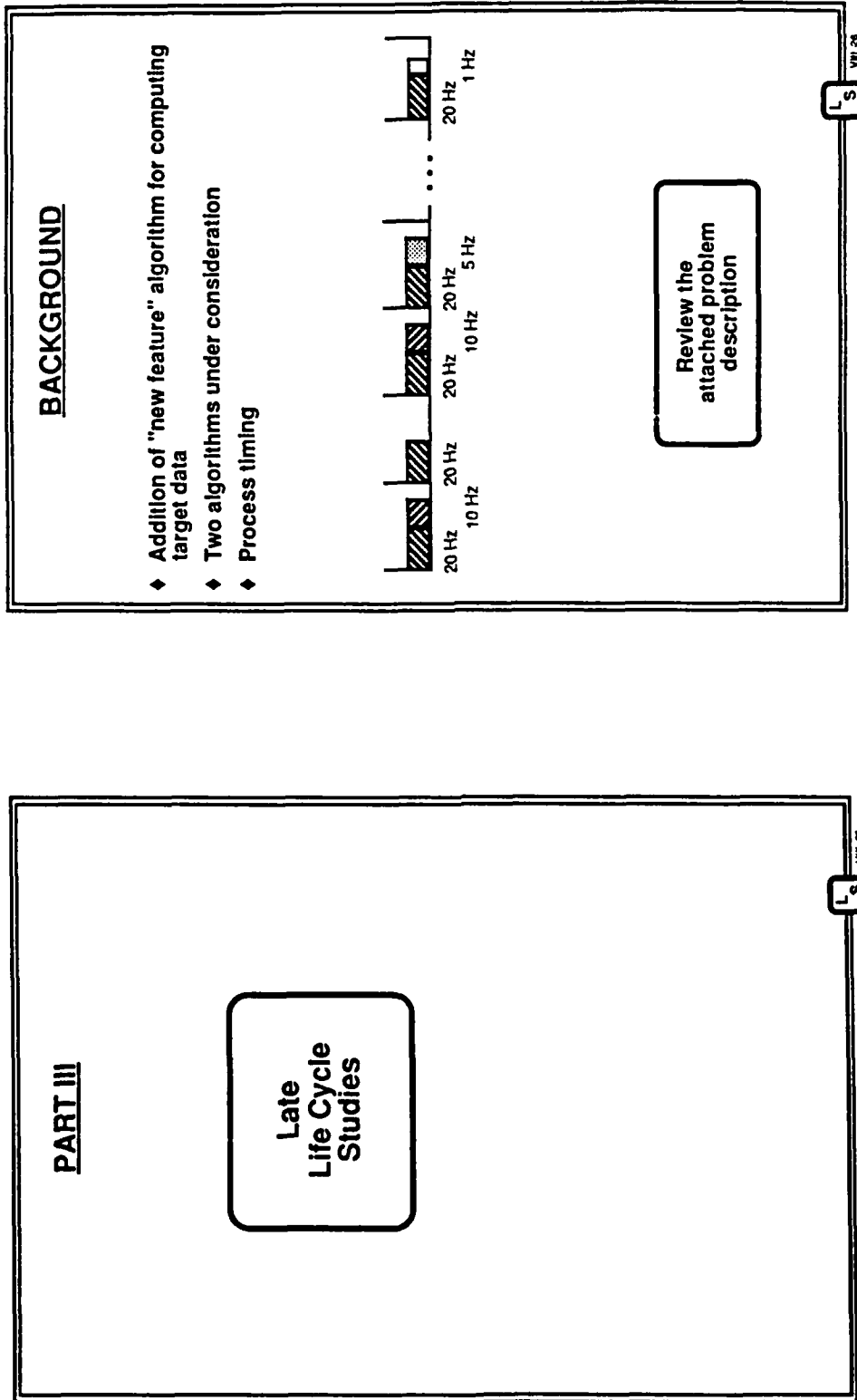
L S VII 24

VIII - 12

Notes

15

Life Cycle Examples



VIII - 13

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

F/A-18 Algorithm Analysis

The following case study information was prepared by a member of the F/A-18 design team. It demonstrates the type of information that can be obtained for SPE studies, and the approach he took in gathering the data. We are grateful for the effort he put forth to support this project.

".... Estimating the word count for each algorithm could be very difficult so I decided to simplify it a bit. I planned to count "high-level" instructions and then use an expansion factor (high-level to assembly) to determine the total number of assembly language instructions.

Determining a valid expansion factor is the hard part: some high-level instructions can be represented as single assembly instructions, but most require two or more instructions. Eventually I came up with the following method:

Assignment	Load (e.g. LD), Store (e.g. SD)
Add/Subtract	Load, Add (e.g.) AD, Rescale (e.g. LALD), Store (1)
Multiply	Load, Multiply (e.g. MDR), Rescale, Store
Divide	Load, Rescale, Divide (e.g. D), Store

Note that the example instructions are for double precision (32-bit) integer arithmetic. The divide is not double precision: Single precision divide (it's 4x as fast). The AYK-14 XN-5 has no floating point unit (i.e. fixed point arithmetic is used throughout).

To "validate" the expansion factor I used data from a previous project. This implemented an algorithm which was first modelled in Fortran. The Fortran statements were classified and counted:

	High-Level	Est Asm	
Assignment	59	118	
Add/Subtract	57	228	
Multiply	28	112	
Divide	9	36	
Total	153	494	--> 3.22 Est Expansion

The project actually used 422 assembly language instructions (sorry - no break down into catagories) which results in an 2.76 expansion factor. Because the project concentrated on optimizing memory usage, I think that 2.75 is a little low for the average project. I believe the original assumption (1) will work.

The expansion factor mentioned from high-level statements to assembly level statements does not account for instructions which require two machine words. These words don't degrade execution speed (any more than the rates listed below) but they do take up more memory. In general, there is a 20%-50% increase from instruction count to memory requirements (e.g. 10 instructions may take 12 to 15 machine words).

Estimated execution performance for the assembly language statements is listed below (in microseconds):

INSTR	DESCRIPTION	XN-5	XN-6
LD	Load Double	2.49	0.95
SD	Store Double	2.68	2.10
AD	Add Double	2.73	1.19
LALD	Left Shift Db1	1.89	1.11
MD	Multiply Double	8.27	4.07
D	Divide	9.87	4.38 (note single precision)
L	Load Single	2.24	0.80
S	Store Single	1.86	1.15
A	Add Single	2.24	0.94
LALS	Left Shift Sngl	1.47	0.90
M	Multiply Single	5.40	2.19

The Algorithms -

Both algorithms will require an interface to the current program. This interface (setting data up etc.) has been estimated to require 1000 assembly words. No mix of statements has been given so I had to guess. My guess is the result of looking at an algorithm intended to function similarly to the two candidate algorithms. I counted the mix of high-level statements; this should be used to determine the overall makeup of this interface. I would assume that all of these instructions run during each pass.

	HOL	%total	
Assignment	37	37%	
Add/Subtract	20	20%	(2)

Multiply	31	31%
Divide	12	12%
Total	100	

This interface will run at a 20Hz rate. It sets data up to be used by the algorithms by "filtering" 20Hz data to be used by the algorithms at 1Hz. Data is passed from the 20Hz task to the 1Hz task with the use of common memory.

Both algorithms are called only in certain conditions: the aircraft "master mode" must be A/A; the radar must be in "angle-only" track; and the radar must not have range information available. Essentially, the pilot only has angular data (position, rates, acceleration) to work with.

The calling conditions for the algorithm take it out of the "worst-case" execution path (which would be "Track While Scan" (TWS) mode with 10 targets). In the future, TWS mode will have angle-only track data for up to ten targets; if we tried to use the algorithm on all the targets, this would be the worst case path. I think we should use this as a separate case within the study. This case would require that the "interface" and the 1Hz task execute for each target.

Initially I thought that TWS would have a looping structure to handle up to ten targets. This looping structure actually is in the Tactical Controls and Displays (TCD) module. The looping structure is required for the display of the target data. The A/A module apparently doesn't compute the missile launch parameters for each target - only for the Lock and Shoot (L&S) target. This makes modelling the A/A module straight forward: it is essentially a straight through path (see the flow chart). Following are some details (I have omitted most details):

Block	Assign	Add/Sub	Mult	Div	
B.1	860	0	0	0	(mostly setup logic)
B.2	317	172	266	103	(matrix transforms)
B.3	0	0	0	0	(new 20Hz goes here)
B.4	340	184	285	110	(missile calculations)
B.5	363	196	304	118	(launch zone calcs)
B.6	272	147	228	88	(missile calcs)
B.7	640	0	0	0	(mostly logic)

NOTE: The above instructions are single precision, not double precision.

ALGORITHM #1

I had the best data for algorithm #1 (because it was developed "in-house"). I was able to get a Fortran listing of the model and some additional data. Because the algorithm has an internal looping structure, I decided to "flowchart" the algorithm and then detail the number of high-level statements in each "block". The statement counts are listed below (a flowchart should be attached).

Block	Assign	Add/Sub	Mult	Div
-------	--------	---------	------	-----

A.1	20	0	5	3	
A.2	15	17	1	5	
A.3	see numbers for EKF "subroutine"				
A.4	79	54	110	8	
A.5	7	0	0	0	
E.1	7	3	4	2	
E.2	19	37	39	0	
E.3	14	0	0	0	
E.4	28	56	14	0	
E.5	27	17	25	2	
Total	216	184	198	20	==> 618 "high-level"
	432	736	792	80	==> 2040 assembly language

Note: The algorithm requires (at least) 282 Data Words

The flowcharts for algorithm #1 indicate a loop which is bounded by NINT. NINT is the number of "integration" steps which are used by the Kalman filter. The number of integration steps is determined by the maximum integration step size and the length of time which must be integrated over. For this implementation NINT will be 2.

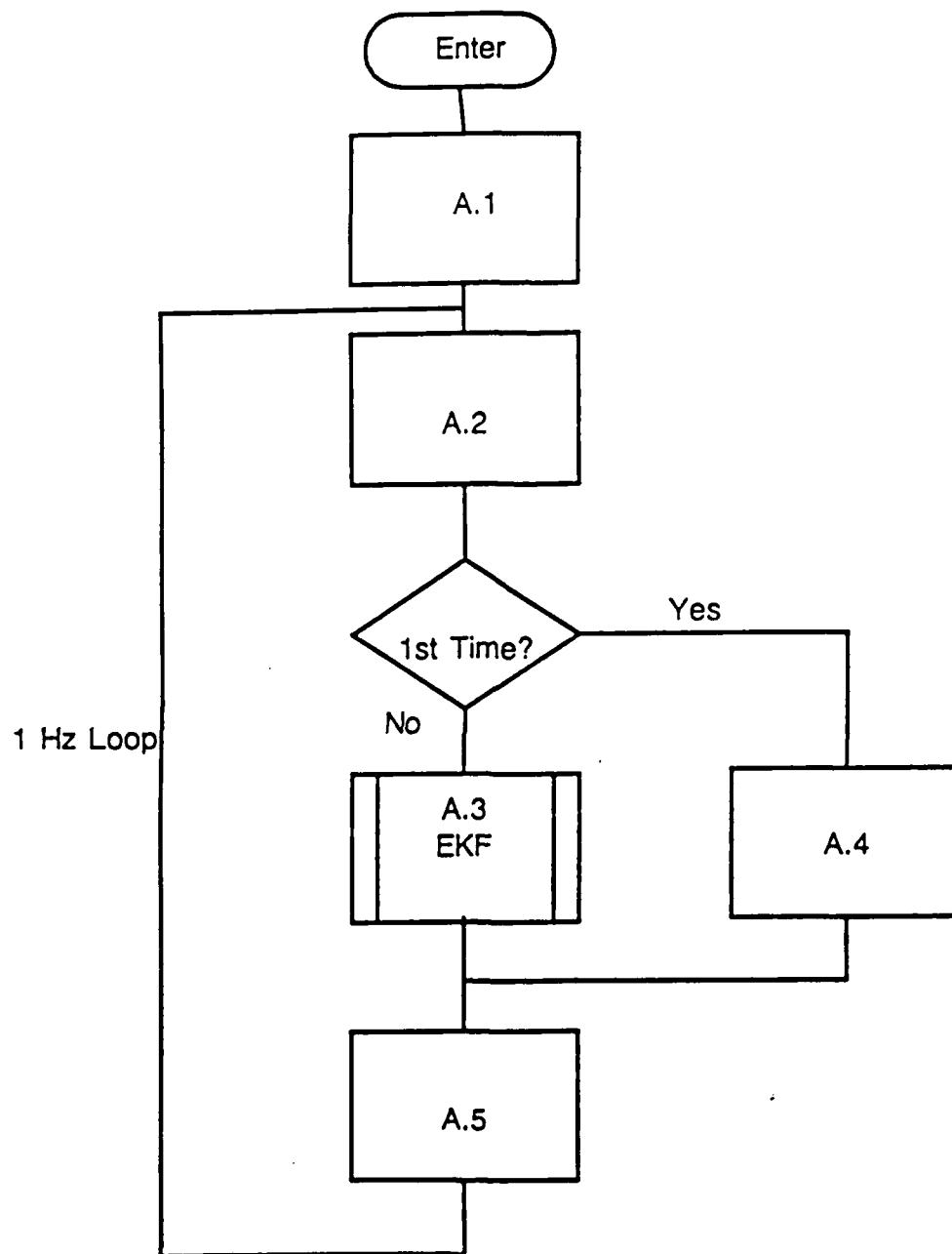
ALGORITHM #2

Algorithm #2 has little data (basically taken from past verbal discussions). The algorithm uses the same basic method used in past algorithm implementations with a little better handling of initial data.

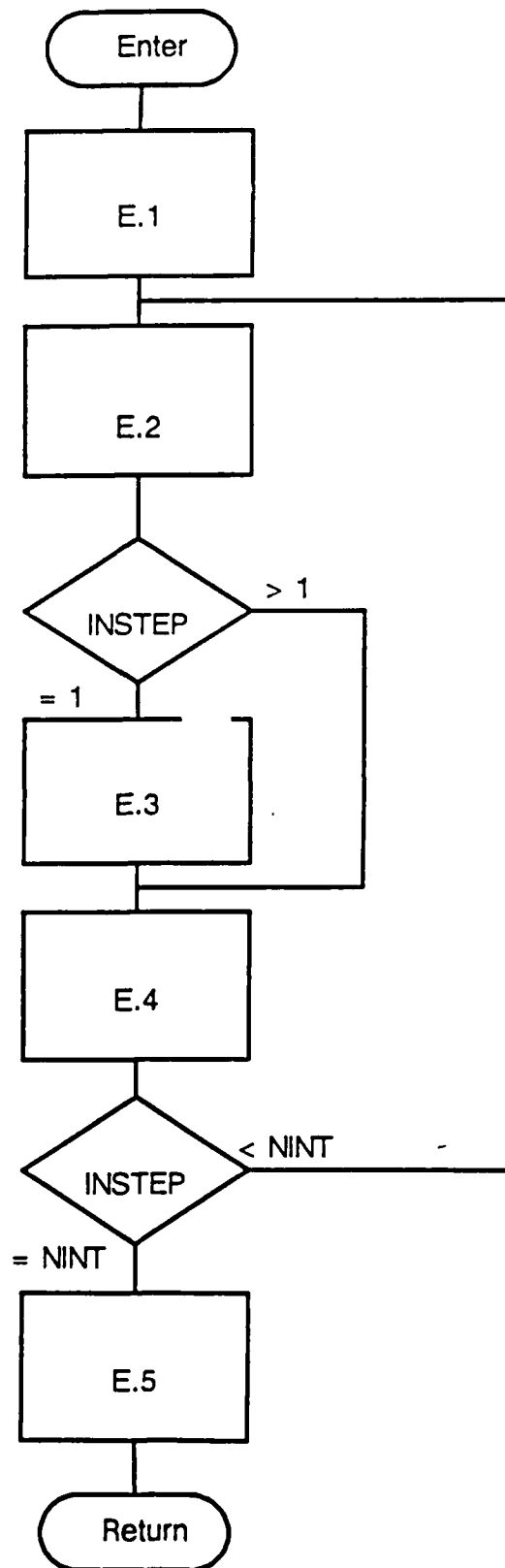
The algorithm's worst case path is also it's steady state path (at least it is very close to being the worst case path). The worst case path will be close to 1000 assembly statements long. Again, we have no mix of the statements so we must assume the percentage mix outlined above (2).

FINAL NOTE:

Algorithm #1 will be implemented in future OFP releases. The rational for doing this is performance of the algorithm in various non-optimum situations. Execution time and algorithm size were not much of an issue (since both algorithms have the same estimated order of magnitude of execution time and size).



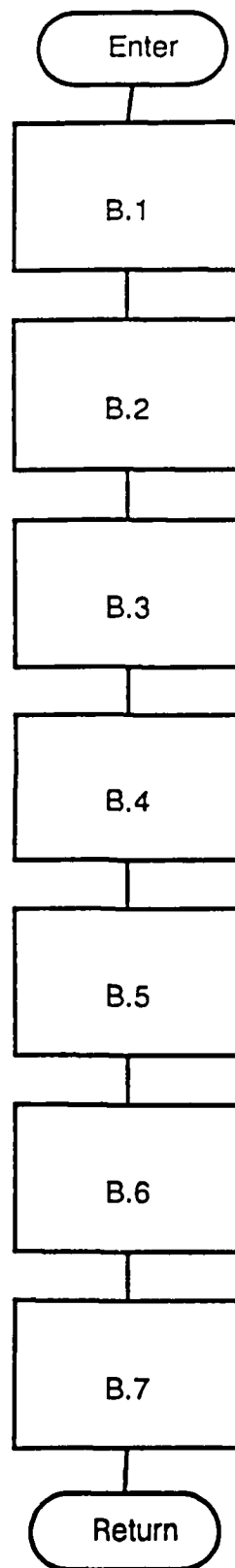
Top Level - Algorithm #1



EKF
Subroutine
Algorithm #1

Block B.3 is where
the 20Hz interface
will reside.

NOTE: There is no
looping structure which
I thought was in TWS.



A/A Processing Flowchart

[illegible]

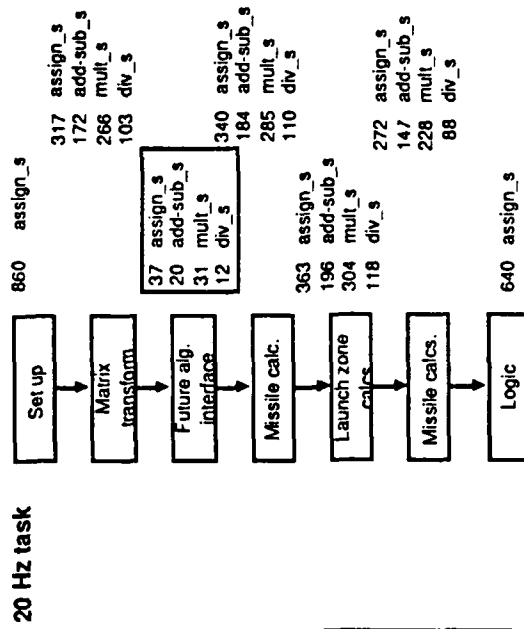
Life Cycle Examples

MODEL FORMULATION

- ◆ Workloads?
- ◆ Configuration?
Devices
Processing rate/time
- ◆ Software execution?
- ◆ Resource requirements?

L S VII 27

SOFTWARE MODELS



L S VII 28

VIII - 14

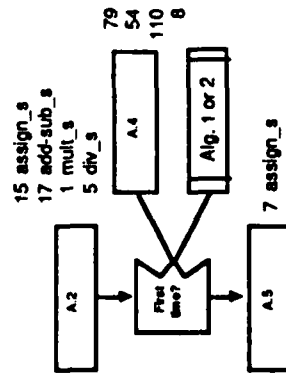
© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

[illegible]

Life Cycle Examples

NEW SOFTWARE

New 1 Hz task



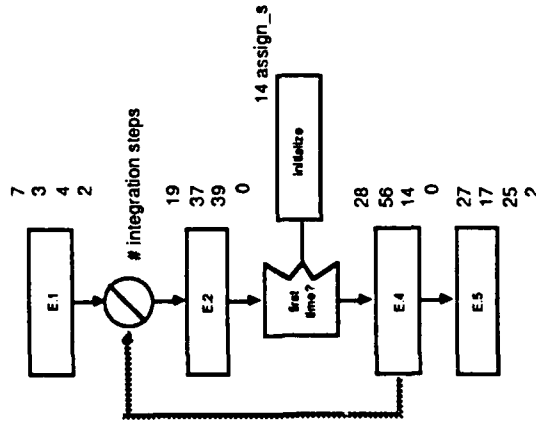
Algorithm 2

Summary
370 assign_s
200 add-sub_s
310 mult_s
120 div_s

L S VII 28

NEW SOFTWARE

Algorithm 1



L S VII 30

VIII - 15

A full page of graph paper with a uniform grid of small squares. The grid consists of 20 columns and 20 rows. A thicker vertical line runs down the right side of the page, approximately one-fifth of the way from the right edge. In the top right corner, there is a small rectangular box containing the number "18". The entire page is enclosed in a double-line border.

LS

Life Cycle Examples

ALGORITHM COMPARISON

Edit the SDF named fa18_init. Solve the baseline model. Then make the changes necessary to evaluate the two algorithm alternatives. What are your conclusions?

L S
VR 31

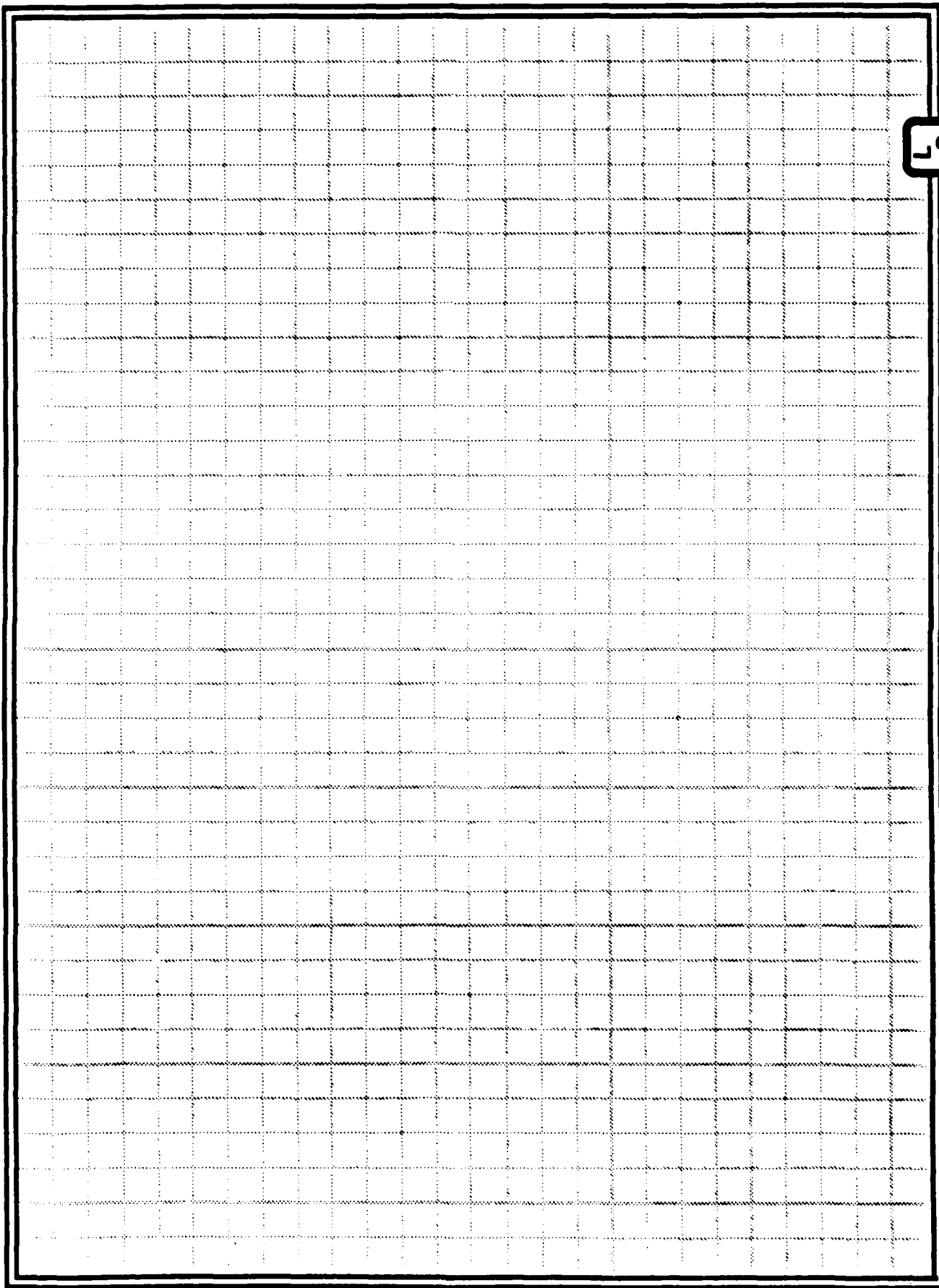
OTHER LATE LIFE CYCLE STUDIES

- ◆ Compare growth of software for 83X, 85X, and 87X releases on the AN/AK-14 model XN-5
- ◆ Evaluate exchange of algorithm for ballistic trajectory
- ◆ Addition of "new feature" algorithm for computing target data
- ◆ Evaluate hardware upgrade: XN-5 to XN-6
- ◆ Concurrent processing: single vs. dual executive on XN-6
- ◆ Determine processing frequency for new functions

L S
VR 32

VIII - 16

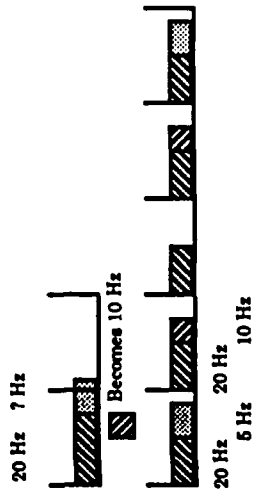
Notes



2

Life Cycle Examples

DETERMINE PROCESSING FREQUENCY



NEED: Easier specifications
Visual results

L S VII 33

SUMMARY

- ♦ Reviewed role of modeling in early mid, & late life cycle stages
- ♦ Reviewed actual case studies
- ♦ Models are similar in all life cycle stages
 - ❖ Level of detail changes
 - ❖ Add advanced model features
- ♦ Next, brief introduction to advanced system models

L S VII 34

VIII - 17

Introduction to Advanced Models

L^s

Introduction to Advanced Models

OBJECTIVES

- Review SPE Modeling Requirements
- Cover strategies for using analytic models to approximate performance
- Introduce modeling techniques for
 - Priorities
 - Maximum concurrent users
 - Communication
 - Synchronization
 - Serialization delays
- Cover POD advanced modeling features
- Discuss future needs

L S IX.1

MODELING REQUIREMENTS

- ◆ Low overhead
- ◆ Accommodate:
 - Incomplete definitions
 - Imprecise performance specs
 - Changes during development

GOAL: *Initially*

- Distinguish between "good" and "bad"

Later

- Increase precision of predictions
- Provide implementation decision support
- Identify problem areas
- Provide guidance for testing

L S IX.2

IX - 1

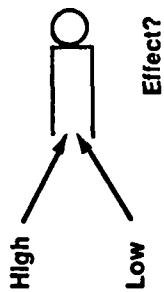
This image shows a full page of blank graph paper. The grid consists of small squares formed by thin, light gray lines. A thicker black border runs along the edges of the page. In the top right corner, there is a small white rectangular label with a black outline containing the number "6". The rest of the page is empty except for the grid pattern.

Introduction to Advanced Models

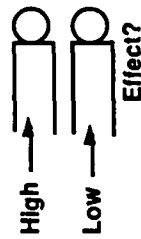
PRIORITY SCHEDULING

- > FAA Processing elements have priorities

- > Simple models ignore priority



- > Alternative



L S IX-3

PRIORITY SCHEDULING

- > Shadow server method



Proportionally slower processor based on utilization of higher priority server

- > How do you know utilization?
- Open models Util = Tput x Serv Time

Closed models:

Estimate, solve, adjust, repeat
Initial guess?

L S IX-4

IX - 2

[illegible]

Introduction to Advanced Models

POD PRIORITY SPECIFICATION

WORKLOAD <name>
 JOB_STREAM = <job name>
 (PRIORITY = <number>)

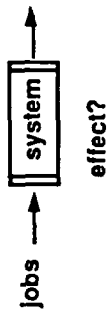
.
 .
 .

0 ≤ <number> ≤ 14
 0 = lowest priority
 parentheses required

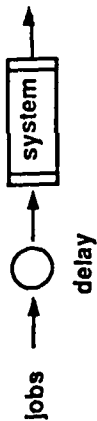
L S R 1

MAXIMUM CONCURRENT USERS

> Simple models ignore constraints



> Alternative



> How do you know the delay?

L S R 1

IX - 3

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

A large rectangular area filled with a fine grid of small squares, typical of graph paper. The grid is composed of horizontal and vertical lines forming a uniform pattern across the page.

15

Introduction to Advanced Models

POD MAXIMUM CONCURRENT USERS

WORKLOAD <name>

MPL = <number>

.
. .
. . .

Terminology - Maximum multiprogramming level

Reports:

Response time - Includes delay

Residence time - within system

POD documentation refers to memory constraints

Technique applies anytime concurrency limit exists

L S α.7

CONCURRENT PROCESSING

- > Apparent vs. real concurrency
- > Specify resource usage per processor
- > Performance concerns
 - Interprocess communication
 - Differs within & between processor
 - Synchronization of processes
 - Trigger sibling processes

L S α.8

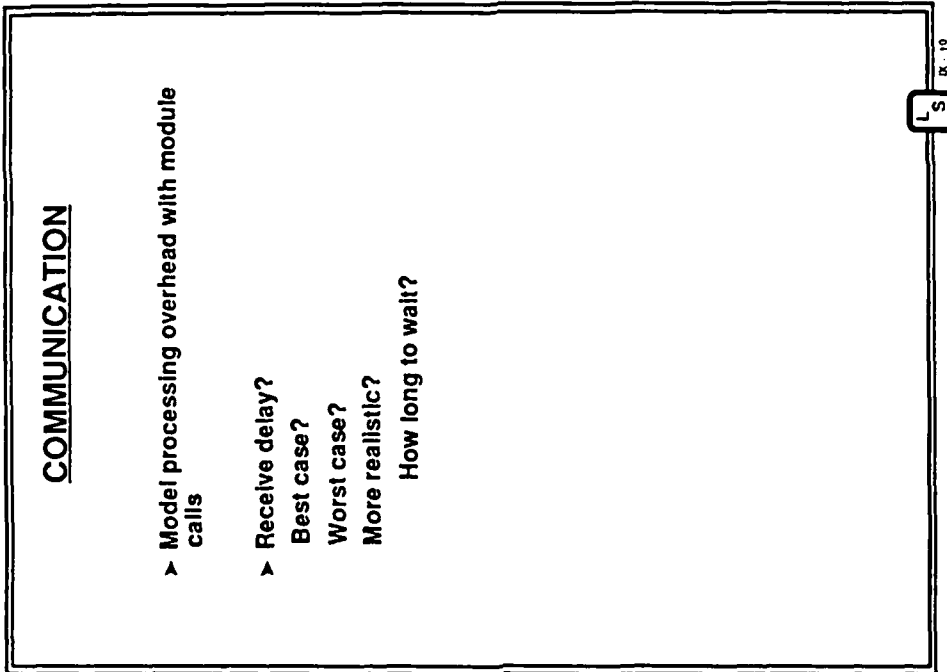
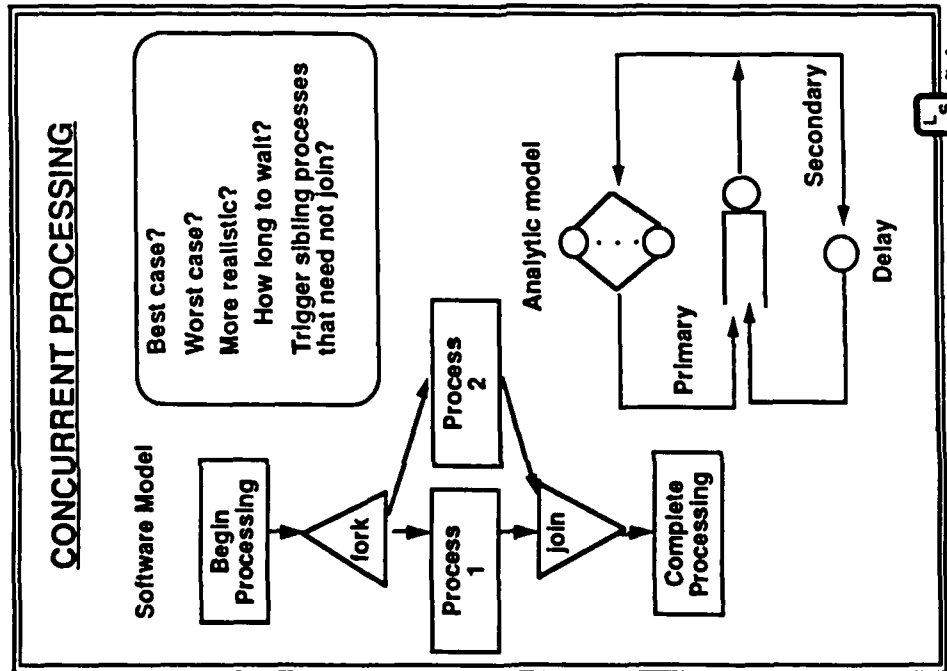
IX - 4

Notes

A large rectangular grid for taking notes, consisting of many small squares. The grid is enclosed in a double-line border. The grid is approximately 20 squares wide and 30 squares high.

19

Introduction to Advanced Models



IX - 5

Notes

LS

Introduction to Advanced Models

POD FEATURES

MODULE <name>
 TRIGGER <identifier>
 (NODE = <device name>
 JOB = <job name>
 WORKLOAD = <wk name>
 DELAY = <time>)
 WAIT (<identifier>) & optional

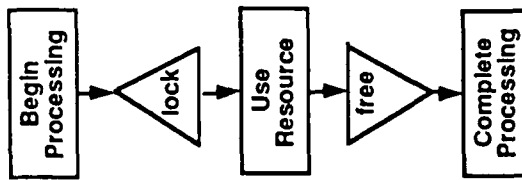
> Delay is msec to send message to node
 > Other varieties of WAIT

L S α . 11

SERIALIZATION DELAYS

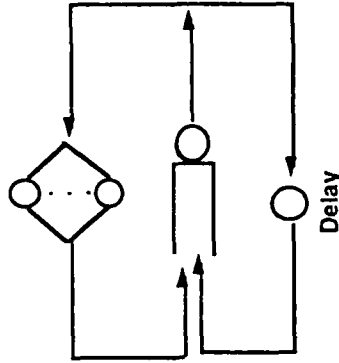
> Passive resource delays
 example - lock, free

Software Model



Best case?
 Worst case?
 More realistic?
 How long to wait?

Analytic model



L S α . 12

IX - 6

Notes

LS

Introduction to Advanced Models

OTHER POD FEATURES

File accesses and device assignments

Database accesses

Memory constraints

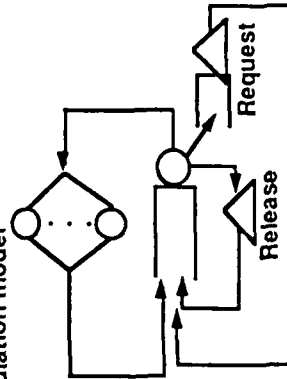
Paging overhead

L S IX - 13

OTHER MODELS

- > When?
- Average values Insufficient
- Complex combinations of advanced features
- More precise predictions
- > Supplemental techniques
- Simulation
- Petri net models
- > Match modeling effort to data precision
- > Want to reuse models, not rebuild them

Simulation model



L S IX - 14

IX - 7

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

Notes

LS

Introduction to Advanced Models

SUMMARY

- > General methods for approximate analytical models
 - Approximations
 - Shadow servers
 - Surrogate delay servers
 - Iterative solutions
- > Use approximations to identify situations that warrant further study
- > Supplement with other modeling approaches when appropriate
- > Use analytic results to double check simulations
- > CAUTION -- models only reflect behavior you define
- > Verification and validation of models are essential

LS
IX - 13

IX - 8

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

General Principles for Performance- Oriented Design

LS

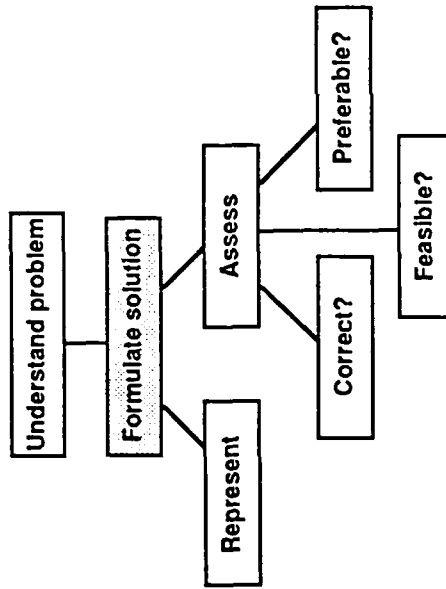
General Principles for Performance-oriented Design

OBJECTIVES

- > Formal procedures for building-in performance
- > Early life cycle principles: Identify requirements and functional architectures with acceptable performance
- > Later life cycle principles
- > Code tuning
- > Case study

L_S X. 1

ENGINEERING DESIGN PROCESS



L_S X. 2

X - 1

[illegible]

General Principles for Performance-oriented Design

MOTIVATION

G.K. Chestertons Observation:

I have seen the truth and
it makes no sense.

Transfer expert knowledge?

L S x 3

REFERENCES

Program Efficiency

Bentley

Knuth

Ferrari

System Design Hints

Lampson

Kopetz

Principle Definition & Application

Smith

Different Perspectives

Large Systems	Programs
Creation	Tuning
Responsiveness	Efficiency

L S x 4

X - 2

Notes

L
S

General Principles for Performance-oriented Design

CENTERING PRINCIPLE

*Identify key scenarios.
Specialize (minimize) processing
of "typical" functions.
Ensure that "worst case" performance
is adequate..*

Typical functions 80-20 Rule

Examples?

L S X . 9

FIXING POINT PRINCIPLE

Actions - Instructions
Information - Data
Point in time - Early vs. Late

*For responsiveness, fixing should be
at the earliest point in time
such that retaining fixing is cost-effective.*

Tradeoff: Flexibility vs. Responsiveness
Compromise: Typical functions

Number targets / tracks to compute
to display

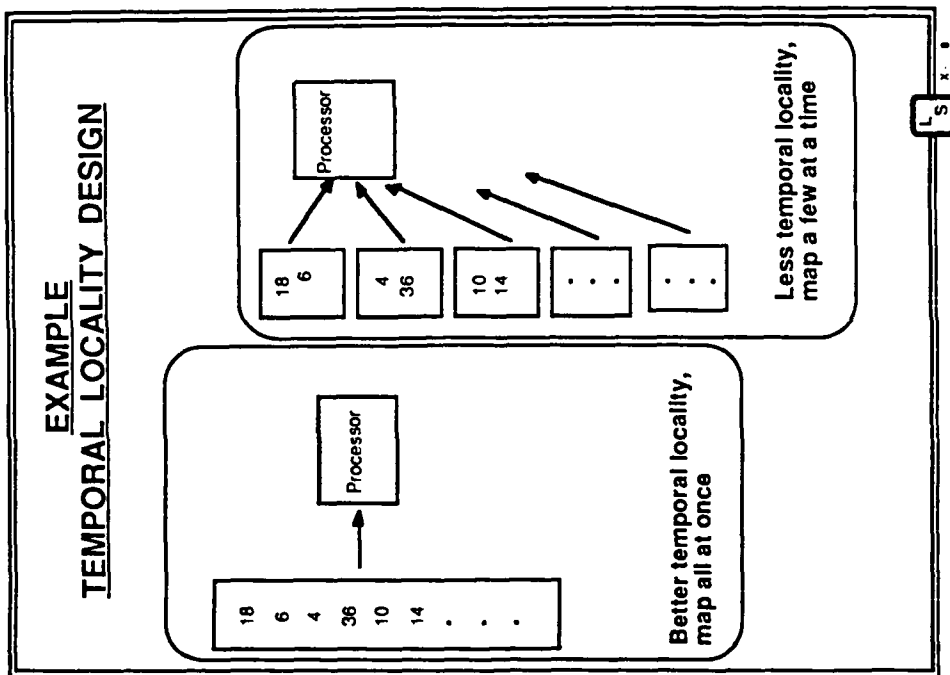
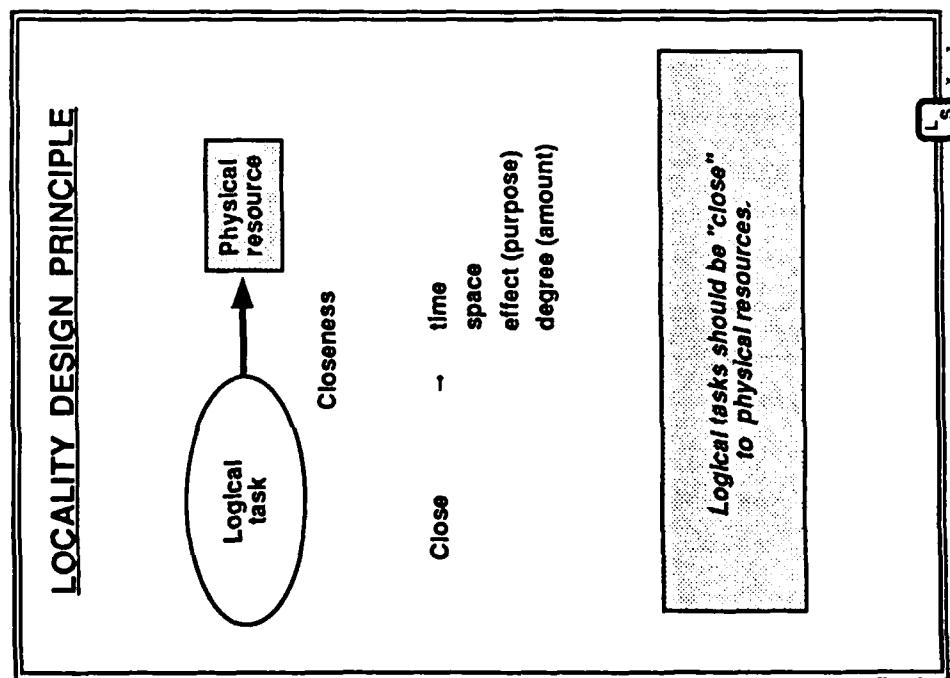
Others?

L S X . 1

X - 3

[illegible]

General Principles for Performance-oriented Design

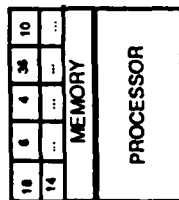


Notes

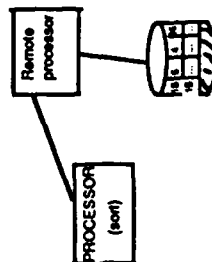
L's

General Principles for Performance-oriented Design

EXAMPLE SPATIAL LOCALITY DESIGN



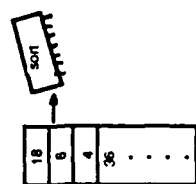
Better spatial locality: In processor's local memory



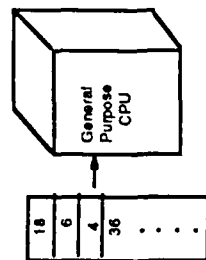
Less spatial locality: on disk attached to remote processor

Signal processing example - frames in global memory, filter coefficients in local memory

EXAMPLE EFFECTUAL LOCALITY DESIGN



Better effectual locality: special purpose processor



Less effectual locality: general purpose processor

Example - custom designed signal processing chip to produce filtered images

L S X 10

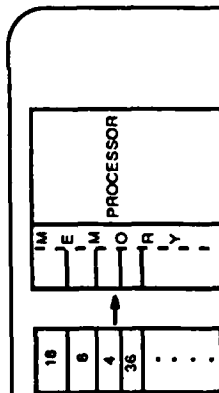
X - 5

Notes

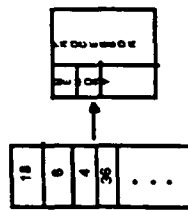
LS

General Principles for Performance-oriented Design

EXAMPLE DEGREE LOCALITY DESIGN



Better degree locality:
size of task matches processor



Less degree locality:
processor is much smaller

Example - timing problems -
so much to do, so little time

PROCESSING VS. FREQUENCY TRADEOFF PRINCIPLE

Trade amount of processing vs. how often

Minimize the processing x frequency product.

- ◆ Consider the number of times a component executes and the amount of processing per execution
- ◆ Minimize both if possible, or at least the product

L S X 12

X - 6

Notes

L S

General Principles for Performance-oriented Design

EXAMPLES PROCESSING VS. FREQUENCY

◆ Architecture:

Export tracks

All in 1 batch
or 1 at a time

◆ Design:

How to implement
batches?

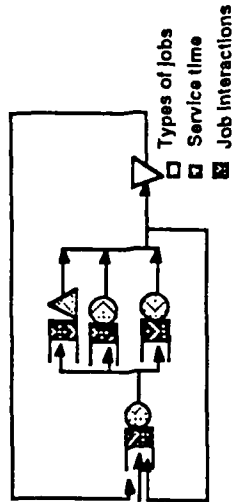
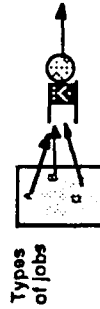
High bandwidth
interface

or internal 'trap' to
call one-at-a-time
routine many times

Others?

L S X 13

INDEPENDENT VS. SYNERGISTIC PRINCIPLES



- ◆ Independent principles optimize performance by reducing resource requirements of software
- ◆ Synergistic principles optimize performance by manipulating patterns of resource requests to reduce contention with other software

L S X 14

X - 7

Notes

L S

General Principles for Performance-oriented Design

SHARED RESOURCES PRINCIPLE

*Resources should be shared when possible.
When exclusive access is required,
minimize holding time + scheduling time.*

FAA key info backups:

All at once

vs.

Fewer at a time, but backups occur
more frequently

L S X 13

PARALLEL PROCESSING PRINCIPLE

*Do processing in parallel only
when speed-up offsets
communication overhead and
resource contention delays.*

Signal processing example:

Alternatives?

How many processors?

L S X 10

X - 8

Notes

LS

General Principles for Performance-oriented Design

INSTRUMENTING PRINCIPLE

Difference between
Instrument, Measure, & Analyze

*Systems should be instrumented to enable
measurement and analysis of
workload scenarios, resource requirements,
and performance goal attainment.*

You cannot control what you cannot measure!

L S X 17

PART II

Later Life Cycle Techniques

- ♦ Same principles but guidance is more specific
- ♦ Locality design may exploit hardware features at cost of portability

L S X 18

X - 9

This is a full-page image of a blank sheet of graph paper. The page is covered by a uniform grid of small squares, created by thin, dotted horizontal and vertical lines. This grid is enclosed within a prominent, thick black solid border. In the upper right-hand corner, outside the main grid area, there is a small white rectangular label with a black outline, which contains the number "2". The rest of the page is empty, showing no handwriting or other markings.

LS

General Principles for Performance-oriented Design

DATA STRUCTURE IMPROVEMENTS

Trading space for time

- ◆ Augment data structures
Bit with status (yes or no)
vs. Array of pointers (to yes's)
- ◆ Store precomputed results
vs. Evaluate when needed
- ◆ "Cache" data - Data accessed most often should
be cheapest to access
Tables ordered by likelihood of match

- ◆ "Hardwire" code
Procedures inline
If statements in code vs. table-driven specs
Image in frame buffer vs. draw when needed

Trading time for space

- ◆ Pack (encode) data - Increase time to store/retrieve
- ◆ Opposite of space for time

Ref: Bentley - Writing Efficient Programs

L S X 19

CODE IMPROVEMENTS - INNER LOOPS

- ◆ Table sentinel
Put desired item into end of table
Avoids check for end of table
- ◆ Loop combining & unrolling - *Careful!*
- ◆ Strength reduction - *Careful!*
Test $\text{SQRT}(x) > 100$
vs. Test $x > 10$ (when $x > 0$)
- ◆ Reorder tests - most likely first
- ◆ Exploit common cases
Handle all cases correctly, common cases
efficiently
Border pixels

L S X 20

X - 10

Notes

19

General Principles for Performance-oriented Design

PART III

Tuning:
Improving performance
of existing software

L S X 21

PROBLEM ISOLATION

- ♦ Which programs?
 - System accounting data
 - System software monitors
 - Ad hoc techniques
- ♦ What parts of program?
 - Program sampling monitor
 - Language trace features
 - Language auditing features

L S X 22

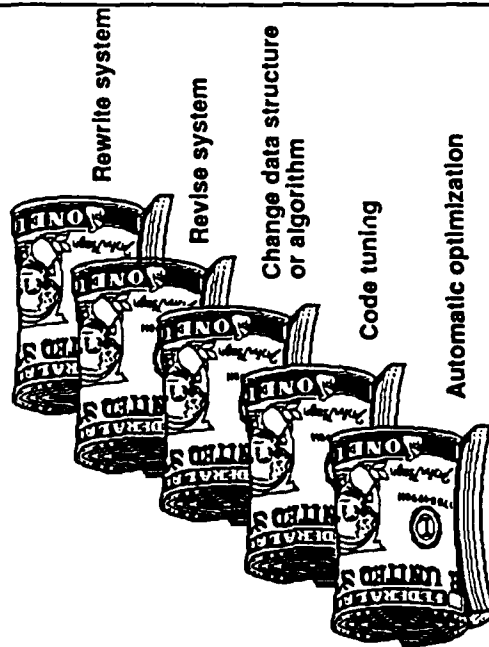
X - 11

Notes

LS

General Principles for Performance-oriented Design

PROGRAM TUNING OPTIONS



Example

Business Application

- ◆ Amazing
- ◆ Inner loop problem is also real time system issue
- ◆ Widespread problem, can you help?

L S X 24

X - 12

Notes

2

General Principles for Performance-oriented Design

PROBLEM DESCRIPTION

- ◆ Performance problem:
Response time complaints
Periodically worse
CPU primary bottleneck
- ◆ Problem Identification:
<many>
1 program consistently on 'top users'
list - both CPU and I/O
- ◆ Symptoms:
1 run / week
> 7 hrs. CPU time / run
> 14 hrs. elapsed time / run

L
S X 2

PROGRAM CHARACTERISTICS

- ◆ History data retrieval (for audit)
- ◆ Selection on any of 4 different keys
e.g., date
id_no
amt_pd
location
etc.

L
S X 2

X - 13

Notes

L 9

General Principles for Performance-oriented Design

IMPLEMENTATION

```
REPEAT
  READ desired key
  INSERT into table (1-4) in ascending order
UNTIL all keys entered;

REPEAT
  READ history rec;
  IF key1 in table1 THEN select;
    (binary search)
  IF key2 in table2 THEN select;

UNTIL all previous years read;
END
```

Performance specifications?

L S X 27

WHY?

- ♦ Evolution
 - Ad-hoc program
 - Need Increased
 - File size Increased
 - Separation between programming & operations & users
- ♦ Is this a fluke?
- ♦ Correction alternatives?

Constraints → Tune Code

L S X 28

X - 14

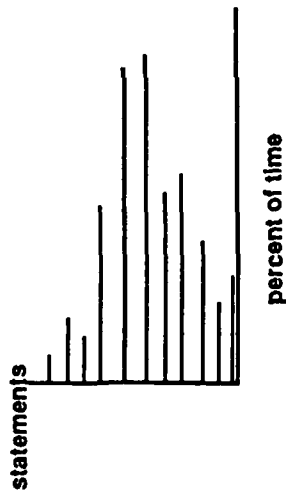
Notes

12

General Principles for Performance-oriented Design

PROBLEM ISOLATION

What parts of program?



L S X 29

PRIMARY LOOP

```

IF rec_key_n = last_wanted
THEN select_it {same as prev. key found?}
ELSE BEGIN
  low := 0; {set initial table bounds}
  high := last + 1;
  REPEAT
    mid := (low + high) * .5; {calc. middle}
    IF rec_key_n = wanted_n(mld) {match middle?}
    THEN select_it {and exit}
    ELSE IF rec_key_n < wanted_n(mld)
    THEN BEGIN {look at lower
      size := mid - low; part of table}
      high := mid + 1;
    END
    ELSE BEGIN {look at upper
      size := high - mid; part of table}
      low := mid;
    END
  UNTIL size = 0; {no match}
  
```

Which part of the code probably corresponds to the largest time spike(s)?

What are some improvement alternatives?

L S X 30

X - 15

Notes

15

General Principles for Performance-oriented Design

REVISED CODE FRAGMENT

```
REPEAT
  IF rec_key_n < wanted_n (Index) THEN
    decrease_Index;
  IF rec_key_n > wanted_n (Index) THEN
    increase_Index;
  (In n) TIMES;
  IF rec_key_n = wanted_n (Index) THEN
    select_it;
```

- ◆ No ELSE's
- ◆ Table size a power of 2
- ◆ No check for previous match
- ◆ No check for match within loop

SUMMARY

- > General principles for performance-oriented design
- > Early & late life cycle applications
- > Examples

L S X 32

X - 16

L S X 31

Notes

15

Integration of SPE

LS

Integration of Software Performance Engineering

OBJECTIVES

- > To review other state of the art and state of the practice techniques for controlling software development
- > To examine how SPE relates to the other areas
- > Review applicability of SPE and POD for MC-ECS
- > Review this technology transfer project
- > Discussion

L S XI - 1

SPE COMPATIBILITY WITH SOFTWARE ENGINEERING

Methodologies

Design principles
Higher level languages
Rapid prototyping

Project Management

Costs & schedules
Team organizations
'Mythical Man Month'
Reviews

Tools

CASE tools
'Environments'
Auto pgm generation

Other

Software metrics
Ergonomics
V & V
QA
Fault tolerance
Reliability

L S XI - 2

XI - 1

Notes

LS

Integration of Software Performance Engineering

TOOLS

- ♦ CASE tools
 - Any that predict performance?
 - Experience?
 - Need both tools and methods
- ♦ Environments
 - Integrated set of tools, implicit methodology
- ♦ Interesting features
 - DWIM
 - History transcripts
 - Color graphics interfaces
 - Spatial data management

Are CASE tools useful for maintenance?

L S XI - 3

MOTIVATION FOR SOFTWARE ENGINEERING TOOLS & METHODS

- ♦ Productivity
- ♦ Correctness
- ♦ Maintainability
- ♦ Reliability
- ♦ Usability
- ♦ Security
- ♦ Testability
- ♦ Efficiency
- ♦ Portability
- ♦ Reusability

Are these compatible with performance?

L S XI - 4

XI - 2

Notes

LS

Integration of Software Performance Engineering

Part II

Using Software
Performance
Engineering

L S XI - 3

EARLY APPROACHES

Small machines → Space & time of programs

Systems of programs:

- ◆ Strict performance requirements:
labor-intensive simulations
- ◆ Other systems:
"fix it later" approach

L S XI - 3

XI - 3

Notes

LS

Integration of Software Performance Engineering

"FIX IT LATER" APPROACH?

Original Premises:

Principle requirements: correct, maintainable
Performance problems are rare
Hardware is (relatively) inexpensive
It's too expensive to build high-performance software
Tuning can be done later

Current Reality:

MANY large systems cannot be used initially due to performance problems. WHY?

Problems are often fundamental architecture or design problems not inefficient code.

Data on time & cost of performance is outdated.

Maintenance of tuned code is expensive.

Performance problems are visible & memorable.

L S XI 7

SPE ALTERNATIVE

- ♦ SPE is sensible and it works
- ♦ Moderate approach between the extremes of the early approaches
- ♦ SPE formalizes the performance techniques that experts use
- ♦ Benefits those with less experience in performance-oriented design
- ♦ Improves performance intuition

L S XI 8

XI - 4

Notes

LS

Integration of Software Performance Engineering

PART II

SPE / POD Project Review

L S XI 10

OVERVIEW

- ♦ Project review
- ♦ Technology transfer discoveries
- ♦ Proposed future directions

L S XI 10

XI - 5

Notes

LS

Integration of Software Performance Engineering

CASE STUDIES CONSIDERED

- ♦ Compare growth of software for 83X, 85X, and 87X releases on the AN/AK-14 model XN-5
- ♦ Evaluate exchange of algorithm for ballistic trajectory
- ♦ Addition of "new feature" algorithm for computing target data
- ♦ Evaluate hardware upgrade: XN-5 to XN-6
- ♦ Concurrent processing: single vs. dual executive on XN-6

Others identified, but insufficient data for near-term case study

Selected algorithm addition because representative, of interest to designers, and sufficient data available

L S XI - 11

APPLICABILITY OF POD & SPE

- ♦ Found many software evaluations that could benefit from the technology
- ♦ Case study had no particular performance problems; the tool confirmed that there should be no surprises
- ♦ POD enables the rapid evaluation of many tradeoffs
 - Applying the algorithm to multiple targets
 - Varying the number of integration steps
 - Examining startup processing as well as steady state
 - Varying the hardware speed

L S XI - 12

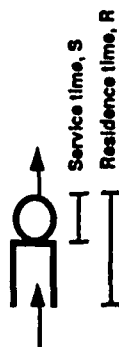
XI - 6

[illegible]

Integration of Software Performance Engineering

TECHNOLOGY TRANSFER DISCOVERIES

- ♦ Usability limitations:
 - NOSC approach is *vital* to technology transfer: Develop prototype into demonstration version
 - Make tool more appealing with visual Interface
- ♦ Theory limitations: Periodic arrivals in queueing theory?



Standard queueing models compute average congestion: $R = (N+1)S$
 If periodic arrivals do not collide: $R = S$
 Determining if collisions occur is nontrivial

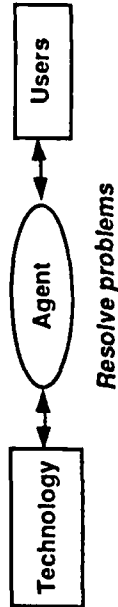
If your only tool is a hammer, every problem looks like a nail

Effective technology transfer
 → Augment tools and SPE methods

LS XI - 13

KEYS TO TECHNOLOGY TRANSFER SUCCESS

1. Transfer agent



2. Clear benefit
3. Compatible with existing methods, experiences, needs
4. Easy to understand, easy to try, measurable results
5. Expert participation

LS XI - 14

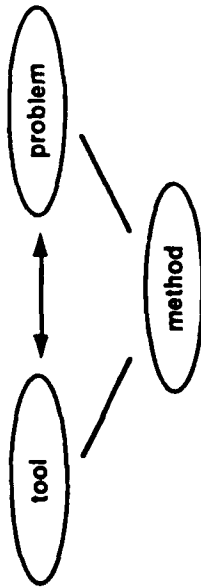
XI - 7

Notes

L S

Integration of Software Performance Engineering

DISCOVERIES



Key problems:

perceived applicability of analytical models
lack of method
ease of use

You cannot address these independently

L S XI - 18

COORDINATED DEVELOPMENT

1. Tool must match the problem
2. Ease of use is vital to acceptance & use
3. Mission critical embedded computer system effectiveness is vital:
Build the *right* system
and
Build the system right

L S XI - 18

XI - 8

Notes

Ls

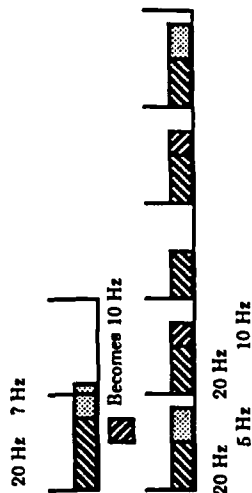
Integration of Software Performance Engineering

PROPOSAL

- ♦ Continue case-study-driven applied research
Select another MC-ECS case study
Apply SPE and POD
Document applicability
Identify and *resolve* limitations in methods / tools
Transfer technology
- ♦ Anticipated enhancements
The following were typical F18 software performance questions and concerns. They are likely to occur in similar systems.

L S XI - 17

DETERMINE PROCESSING FREQUENCY



NEED: Easier specifications
Visual results

L S XI - 18

XI - 9

Notes

L S

Integration of Software Performance Engineering

EVALUATE BUS CONTENTION AND PROCESSOR INTERFERENCE

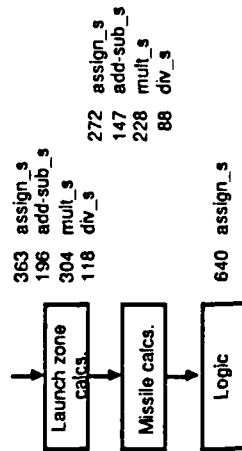
- ◆ Poll devices at beginning of cycle
- ◆ Channel 3 dedicated to handoff protocol between processors
- ◆ Data requirements and algorithms are interrelated

NEED: Case study execution details to construct models
Possible POD extensions

L S XI - 19

EXPLICIT DATA REPRESENTATION

Current POD specifications: processing requirements



- ◆ Data structure alternatives affect # instructions executed
- ◆ Need to know data accesses to determine I/O requirements

NEED: Define data representation and analysis requirements
Tool enhancements

L S XI - 20

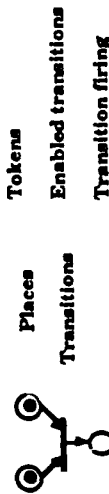
XI - 10

[illegible]

Integration of Software Performance Engineering

INTEGRATE ADDITIONAL PERFORMANCE SOLVERS

- ♦ Periodic job contention
- ♦ Ada rendezvous
- ♦ Synchronization of concurrent processes
- ♦ Data latency
- ♦ Must respond to critical data values within window
- ♦ Data that is only valid for brief window
- ♦ Augment graph analysis and queueing theory with petri nets:



NEED: Case study details
Prototype petri net interface to
POD model definitions

L S XI - 21

IMPROVE POD USABILITY

- ♦ Fine tune reports
- ♦ Graphics feedback
- ♦ Automate typical analysis problems

:

L S XI - 22

XI - 11

Notes

L S

Integration of Software Performance Engineering

INTEGRATE WITH SOFTWARE ENGINEERING ENVIRONMENTS

- ♦ Leverage models with design information
- ♦ Automatically update models
- ♦ Provide performance predictions to designers

NEED: Target software engineering environment
Common language (database)
Integration plan

L S XI - 23

PROJECT SUMMARY

- ♦ Project review
F18 software models
Technology transfer discoveries
Approach
- ♦ Future directions
Applied research
Technology-transfer goals
Future directions suggested by F18 study
Likely to find other areas

Performance of MC_ECS is vital

SPE + POD technology resolves vital problems
Hasten transfer by adapting the technology
to typical (hard) problems

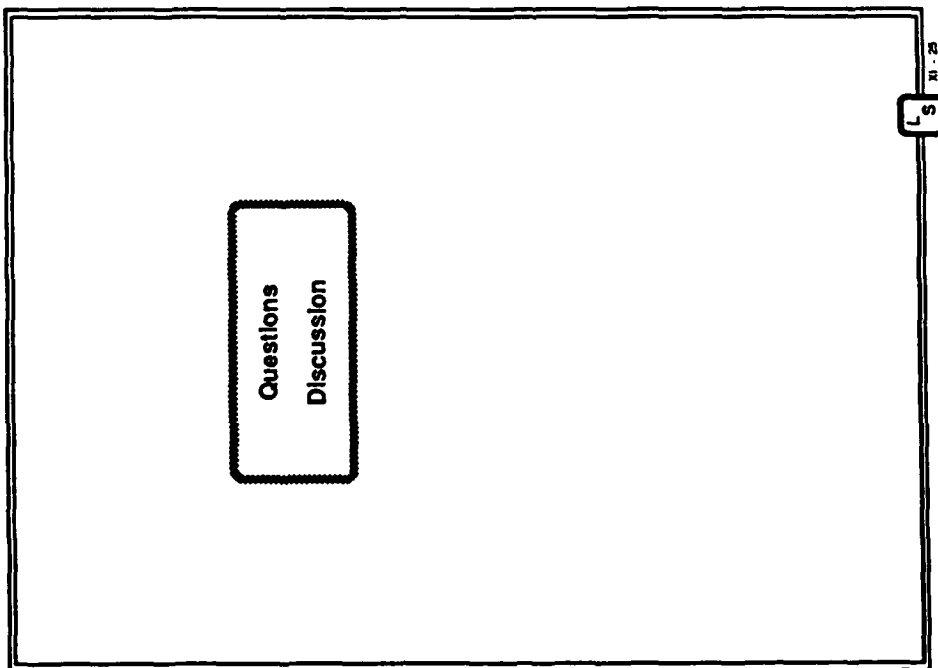
L S XI - 24

XI - 12

Notes

LS

Integration of Software Performance Engineering



XI - 13

© 1989 by Performance Engineering Services Div., L&S Computer Technology, Inc.

References



Performance Engineering Bibliography

Connie U. Smith

Performance engineering is a methodology for constructing software systems to meet performance requirements. It begins early in the software life cycle when requirements and designs are formulated. By modeling and assessing the predicted performance early, one can choose an appropriate functional architecture before significant efforts are invested in an inferior one. Since optimization is performed prior to coding, the maintainability is better (than for code that has been tuned) and the productivity of the development staff is improved (since efforts are not wasted on inferior products.)

This bibliography was created to provide practitioners who are interested in applying the performance engineering methodology with the information they need to do so. It was originally published in the Fall 1985 *CMG Transactions* Special issue on Software Performance Engineering. This version is updated to include new references.

This list focuses on papers reporting experiences with software performance engineering. These papers are identified in the list with the keywords EXPR for recent experience and HIST for experiences prior to 1975. References to other related subjects have also been included: papers describing how to do various PE activities (METH), techniques for modeling software performance (MDLG), tools for performance modeling (TOOL), techniques for measuring software (MEAS), and techniques for improving program performance (PGMS). A new category has been added: design guidelines or principles for high performance software (PRIN).

The bibliography is quite large. To keep it focused, references on the related topics MDLG, MEAS, PGMS, TOOL, are limited. For example, the literature on general queueing network modeling issues is extensive and is not included here. Only a representative set of books on performance evaluation and system modeling are included (those that specifically address software performance.) Readers interested in those topics should refer to the bibliographies previously printed in the *CMG Transactions*.

You should be able to find most academic publications in a university library. A California company provides references for a fee: Information On Demand, PO Box 1370, MS 3395160, Berkeley, CA 94701. CMG, ACM, IEEE publications can be obtained from the respective headquarters (listed below).

[ACM] Association for Computing Machinery, 11 W. 42nd St., New York, NY 10036.

[AGR82] Jon R. Agre, Satish K. Tripathi, "Modeling Reentrant and Non-reentrant Software," *Proc. Conf. on Measurement and Modeling of Computer Systems*, Seattle, Aug. 1982, 163-178. (MDLG)

[ALE82] William Alexander, Richard Brice, "Performance Modeling in the Design Process," *Proc. National Computer Conf.*, Houston, June 1982. (EXPR)

[ALE86] C.T. Alexander, "Performance Engineering: Various Techniques and Tools," *Proc. CMG 86*, Dec. 1986, pp. 264-267. (METH)

[ALL79] F.W. Allen, "A Predictive Performance Evaluation Technique for Information Systems," *Proc. Fourth International Symposium on Modeling and Performance Evaluation of Computer Systems*, Vienna, February 1979. (MDLG)

- [ALL82] Leilani E. Allen, "How to Obtain Accurate Workload Forecasts from Users," *Proc. Computer Measurement Group Conf. XIII*, San Diego, Dec. 1982, 109-118. (METH)
- [ALL83] Karl S. Allen, Allan I. Levy, "A Case Study in Software Performance Engineering," *Proc. Int. Conf. on Computer Capacity Management*, Apr. 1983, 255-266. (EXPR)
- [ALL85] A.O. Allen, "Introduction to Information System Modeling," *Proc. Int. Conf. on Information Mgt.*, April 1985, pp. 152-158. (METH)
- [ALM79] Guy T. Almes and Edward D. Lazowska, "The Behavior of Ethernet-Like Computer Communications Networks," *Proc. 7th ACM Symposium on Operating Systems Principles*, December 1979. (MDLG)
- [AMM83] Reda Ammar, "Software Optimization Using User Models," U. Connecticut Tech. Rpt. CS-83-12, 1983. (METH)
- [AND84] Gordon E. Anderson, "The Coordinated Use of Five Performance Evaluation Methodologies," *CACM*, 27,2, Feb. 1984, 119-125. (EXPR)
- [BAG84a] R. Bagrodia, K.M. Chandy, J. Misra, "May: A Message-Based Simulation Language for Small Computers," *Proceedings of COMPCON*, Fall 1984, Washington, DC, Sept. 1984. (MDLG)
- [BAG84b] R. Bagrodia, "Performance Analysis of Distributed Software Using Message Based Simulation," *Proceedings of CMG XV*, San Francisco, December 1984. (MDLG)
- [BAT76] Bates, D. (ed). "Program Optimization," *Infotech State of the Art Report 30*, Infotech International, Maidenhead 1976. (PGMS)
- [BEI78] Boris Beizer, *Micro-Analysis of Computer System Performance*, Von Nostrand Reinhold, 1978. (MDLG)
- [BEI84] Boris Beizer, "Software Performance," in C.R. Vick, C.V. Ramamoorthy eds., *Handbook of Software Engineering*, Von Nostrand Reinhold, 1984, pp.413-436. (MDLG)
- [BEK85] G.A. Becker, "Capacity Planning for Applications Still Under Development," *Proc. CMG 85*, Dallas, Dec. 1985, pp.700-703. (METH)
- [BEL77] T.E. Bell, D.C. Bixler, M.E. Dyer, "An Extendable Approach to Computer-Aided Software Requirements Engineering," *IEEE Trans. Soft. Eng.*, SE-3,1, January 1977, pp. 49-59. (TOOL)
- [BEL87] T.E. Bell, "Performance Engineering: Doing It 'Later' on Large Projects," *CMG Transactions*, Winter, 1987, pp. 75-83. (EXPR)
- [BEN82] Jon L. Bentley, *Writing Efficient Programs*, Prentice-Hall, 1982. (PGMS)
- [BER74] T. Beretvas, "A Simulation Model Representing the OS/VS2 Release 2 Control Program," *Lecture Notes in Computer Science 16*, Springer Verlag, 1974, 15-29. (HIST)
- [BER84] Margaret E. Berry, "The Best of Both Worlds: An Integrated Approach to Capacity Planning and Software Performance Engineering," *Proc. Computer Measurement Group Conf. XV*, San Francisco, Dec. 1984, 462-466. (EXPR)
- [BGS83] CRYSTAL Modeling Package, BGS Systems, Inc., Waltham, MA., 1983. (TOOL)

- [BOE81] B.W. Boehm, *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, NJ, 1981. (SF ref)
- [BOO82] CMF/MODEL, Boole and Babbage, Inc., Sunnyvale, CA., 1982. (TOOL)
- [BOO79a] T.L. Booth, "Use of Computation Structure Models to Measure Computation Performance," *Proc. Conference on Simulation, Measurement and Modeling of Computer Systems*, Boulder, August 1979. (MDLG)
- [BOO79b] T.L. Booth, "Performance Optimization of Software Systems Processing Information Sequences Modeled by Probabilistic Languages," *IEEE Trans. on Software Eng.*, Vol. SE-5, 1, Jan. 1979, pp. 31-44. (MDLG)
- [BOO80] Taylor L. Booth, Cheryl A. Wiecek, "Performance Abstract Data Types as a Tool in Software Performance Analysis and Design," *IEEE TSE*, SE-6, 2, Mar. 1980, 138-151. (MDLG)
- [BOO81] T.L. Booth, R. Ammar, R. Lenk, "An Instrumentation System to Measure User Performance in Interactive Systems," *Journal of Systems and Software*, 2, 1981, 139-146. (MEAS)
- [BOO86] T.Booth, R. Hart, B. Qin, "High Performance Software," *Proc. Hawaii Int. Conf. Sys. Sciences*, Jan. 1986, pp. 41-52. (EXPR)
- [BOU85] J.P. Bouhana, "Software Systems Instrumentation: An End-to-End View," *Proc. Hawaii Int. Conf. on System Science*, 1, Jan. 1985. (MEAS)
- [BRI80] Richard S. Brice, J. Wayne Anderson, "Performance Evolution in a Large Scale System," *Proc. Computer Performance Evaluation Users Group 16*, Orlando, October 1980, 319-330. (EXPR)
- [BRO75] F.P. Brooks, *The Mythical Man Month*, Addison-Wesley, Reading, MA, 1975.(SE ref.)
- [BRO85] J.C. Browne, D.M. Neuse, Jim Dutton, and Keh-Chiang Yu, "Graphical Programming for Simulation Models of Computer Systems," *Proc. 18th Annual Simulation Symposium*, Tampa, Florida, March, 1985. (TOOL)
- [BUZ81] J.P. Buzen, A.I. Levy et. al., "Predicting Software Performance with CRYSTAL," *Proc. Int. Conf. on Computer Capacity Management*, Chicago, 1981, 275-284. (METH)
- [CAB81a] L.F.Cabrera, "Syntax Oriented Analysis of the Run Time Performance of Programs," Ph.D. Thesis, U. of Calif. Berkeley, Electronics Research Laboratory Memorandum No. UCB/ERLM81/30, 1-187, May 1981. (MEAS)
- [CAB81b] L.F.Cabrera, "On the Efficient Generation of Dynamic Program Profiles," *Proceedings of COMPSAC 81*, Chicago, Illinois, October 1981, pp. 219-224. (MEAS)
- [CAB84] L.F.Cabrera, G.Rodriguez-Galant, "Predicting Performance in UNIX Systems From Portable Workload Estimators Based on the Terminal Probe Method," Report No. UCB/CSD 84/194, University of California, Berkeley, August 1984. To appear in *IEEE Trans. Software Engineering*. (MEAS)
- [CAM68] D.J. Campbell, W.J. Heffner, "Measurement and Analysis of Large Operating Systems During System Development," *Proc. 1968 Fall Joint Computer Conference AFIPS Vol. 37*, AFIPS Press, 1968, 903-914. (HIST)

- [CHA82a] K.M. Chandy, J. Misra, R. Berry, D. Neuse, "The Use of Performance Models in Systematic Design," *Proc. National Computer Conf.*, Houston, June 1982, 251-256. (METH)
- [CHA82b] K.M. Chandy, "Planning Information Systems," *Journal of Capacity Management*, 1, 1, 1982, 9-21. (METH)
- [CHU85] L.W. Chung, "A Large UNIX Application Sizing Methodology," *Proc. CMG 85*, Dallas, Dec. 1985, pp. 80-84. (EXPR)
- [CMG] Computer Measurement Group, 6397 Little River Turnpike, Alexandria, VA 22312, (703) 354-3306.
- [COH74] Cohen, J., and Zuckerman, C., "Two Languages for Estimating Program Efficiency," *Communications of the ACM*, Number 6, Volume 17, June 1974, pp. 301-308. (PGMS)
- [COH77] Cohen, J., and Carpenter, N., "A Language for Inquiring about the Run-Time Behavior of Programs," *Software--Practice and Experience*, Volume 7, September 1977, pp. 261-271. (MEAS)
- [DEN78] P. Denning and J. Buzen, "The Operational Analysis of Queueing Network Models," *Computing Surveys*, 10,3, September, 1978. (MDLG)
- [FAL86] Anne M. Falk, "Minicomputer Performance Engineering: A Case Study," *CMG Transactions*, Summer 1986, pp. 14-21. (EXPR)
- [FER84] V. Fernandes, J.C. Browne, D.M. Neuse, and R. Velpuri, "Some Performance Models of Distributed Systems," *Proc. Computer Measurement Group XV*, San Francisco, December 1984, 30-37. (MDLG)
- [FER78] D. Ferrari, *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, New Jersey 1978, Chapter 9. (METH)
- [FER83a] D. Ferrari, "Software monitors," in A. Ralston, Ed., *Encyclopedia of Computer Science and Engineering*, Second Edition, Van Nostrand Reinhold Co., Florence, KY, 1983, 1362-1364. (MEAS)
- [FER83b] D. Ferrari, G. Serazzi, and A. Zeigner, *Measurement and Tuning of Computer Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983, 540 pp., Chapter 8. (METH)
- [FOU85] R.A. Fournet, "Establishing Information Engineering," *Proc. Int. Conf. on Information Mgt.*, April 1985, pp. 48-65. (METH)
- [FRA85] G.A. Frank, C.U. Smith, J.L. Cuadrado, "Software/Hardware Codesign with an Architecture Design and Assessment System," *Proc. Design Automation Conference*, Las Vegas, 1985. (TOOL)
- [FRA85b] J.L. Franklin, P. Hwang, A. Wrenn, T. Park, "Software Analysis Tools: A Method for Developing Performance Model Inputs," *Proc. CMG 85*, Dallas, Dec. 1985, pp. 400-410. (TOOL)
- [FRI85] L. Jeanne Friedman, "Integrating Performance and Capacity Planning into the Application Design Review Process," *CMG Transactions*, 47, Mar. 1985, pp. 28-29. (METH)
- [GAF85] J.E. Gaffney, "Some Metrics for Assessing the Degree of 'Fit' of Software to its Host Computer," *Proc. CMG 85*, Dallas, Dec. 1985, pp. 434-437. (MDLG)

- [GIR83] P.P.V. Giri, "Functional Design of On-Line Applications for Performance," *Proc. Int. Conf on Computer Capacity Management*, New Orleans, 1983, 147-163. (EXPR)
- [GOD85] W. Godwin, W. Suhler, "A Timing Estimation Method for Large System Software Development," *Proc. CMG 85*, Dallas, Dec. 1985, pp. 438-444. (TOOL)
- [GOL78] R.P. Goldberg, A.I. Levy, H.S. Schwenk, "Analysis of an Automated Bibliographic Retrieval System," *Proc. Computer Perf. Evaluation User's Group*, Boston, Oct. 1978. (EXPR)
- [GOS85] J. Gostl, I. Greenberg, "An Application of Queueing Theory to the Design of a Message-Switching Computer System," *Communications of the ACM*, 28,5, May 1985, pp. 500-505. (EXPR)
- [GRA73] R.M. Graham, G.J. Clancy, D.B. DeVaney, "A Software Design and Evaluation System" *CACM* 16,2, February 1973. (HIST)
- [GRA82] G.S. Graham, E. D. Lazowska and K.C. Sevcik: Components of Software Packages for the Solution of Queueing Network Models. *Proc. CPEUG Conference*, October 1982. (TOOL)
- [HAR84] G. Haring, W. Konvicka, "Capacity Planning for an Electronic Mail System," *Proc. European Computer Measurement Association - 12*, Munich, Oct. 1984, 119-125. (EXPR)
- [IEEE] Computer Society of the IEEE, P.O. Box 04699, Los Angeles, CA 90051.
- [ING82] Frank T. Ingrassia, "The Role of Capacity Planning in Software Development," *Proc. Computer Measurement Group XIII*, San Diego, Dec. 1982, 226-235. (EXPR)
- [ING85] Frank T. Ingrassia, "Modeling the Performance of New Online Systems," *Proc. CMG 85*, Dallas, Dec. 1985, pp. 346-357. (METH)
- [IRA83] Performance Analyst Workbench System (PAWS), Information Research Associates, Austin, TX., 1983. (TOOL)
- [JAL77] Jalics, P. J., "Improving performance the easy way," *Datamation* 23,4, April 1977, 135-148. (PGMS)
- [JEN82] S.F. Jennings, A.E. Oldehoeft, "An Analysis of Program Execution on a Recursive Stream-Oriented Data Flow Architecture," *Proc. Hawaii International Conference on System Sciences*, Jan. 1982, 178-187. (MDLG)
- [JOU83] William Jouris, "Introduction to Tuning Batch Applications," *Proc. Computer Measurement Group Conf. XIV*, Washington, Dec. 1983, 371-373. (PGMS)
- [KEL74] J.C. Kelly, "The Theory of Repetition Networks with Application to Computer Programs," Ph.D. Dissertation, Purdue University, December 1974. (MDLG)
- [KNU68] Knuth, D.E., *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Addison-Wesley, Reading, MA (1968). (PGMS)
- [KOB78] Kobayashi, H. *Modeling and Analysis - An Introduction to System Performance Evaluation Methodology*. Addison-Wesley, Reading, MA 1978. (METH)
- [KOP86] H. Kopetz, "Design Principles for Fault Tolerant Real Time Systems," *Proc. Hawaii Int. Conf. Sys. Sciences*, Jan. 1986, pp. 53-62. (PRIN)

- [LAK86] S. Lakshmi, S. Calo, P. Gupta, "Frame Caching in Menu-Driven Videotex Systems," *Proc. FJCC*, Nov. 1986, pp.655-664. (EXPR)
- [LAK87] M. Seetha Lakshmi. "Performance Engineering of a Videotex System," *Proc. Hawaii Int. Conf. System Sciences*, Jan. 1987. (EXPR)
- [LAM84] B.W. Lampson. Hints for computer system design. *IEEE Software*, Feb. 1984, 11-28. (PRIN)
- [LAS72] E.R. Lassetre, A.L. Scherr, "Modeling the Performance of the OS/360 Time Sharing Option (TSO) in W. Freiberger (ed.), *Statistical Computer Performance Evaluation*, Academic Press, 1972, 57-72. (HIST)
- [LAZ80] Edward D. Lazowska. The Use of Analytic Modeling in System Selection. *Proc. CMG International Conference*, December 1980. (METH)
- [LAZ84a] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice Hall, Inc., Englewood Cliffs, 1984, Chapter 14. (MDLG)
- [LAZ84b] Edward D. Lazowska, John Zahorjan, David R. Cheriton, and Willy Zwaenepoel. File Access Performance of Diskless Workstations. Technical Report 84-06-01, Department of Computer Science, University of Washington, June 1984. Also Report No. STAN-CS-84-1010, Department of Computer Science, Stanford University. Submitted to *ACM Transactions on Computer Systems*. (MDLG)
- [LeM82] Eric LeMer, "MEDOC: A Methodology for Designing and Evaluating Large-Scale Real-time Systems," *Proc. National Computer Conf.*, Houston, June 1982, 263-272. (METH)
- [LEV80] A.I. Levy, "Practical Operational Analysis: An MVS Perspective," *Proc. CMG*, Boston, Dec. 1980. (MDLG)
- [LEV85] A.I. Levy, K.S. Allen, "The Role of Performance Engineering in Applications Development," *Proc. Int. Conf. on Information Management*, Apr. 1985, pp. 9-19, also to appear in *Journal of Capacity Management*, Vol. 3, No. 1, Fall 1985. (METH)
- [LEV86] A.I. Levy, V.C.Soder, "Performance Analysis During Application Design: A Case Study," *Proc. CMG 86*, Dec. 1986, pp. 167-172. (EXPR)
- [LIN86] David S. Lindsey, "Do Fortran Compilers Really Optimize?" *CMG Transactions*, Spring, 1986, pp23-27. (PGMS)
- [McN80] McNeil, M., Tracy, W. "PL/I program efficiency," *SIGPLAN Notices* 15,6 June, 1980, 46-80. (PGMS)
- [MOL82] M.K. Molloy, "Performance Analysis Using Stochastic Petri Nets," *IEEE Trans. Computers*, C-31,9, September 1982, 913-917. (MDLG)
- [MOR86] G.A. Morrison, "Performance for a Large, Complex Application," *Proc. CMG 86*, Dec. 1986, pp. 316-320. (METH)
- [NEU83] D.M. Neuse, J.C. Browne, "Graphical Tools for Software System Performance

Engineering," *Proc. Computer Measurement Group Conf. XIV*, Washington, Dec. 1983, 353-355. (TOOL)

[PAA84] Ronald Paans, "Effect of Response Times on Project Costs in MVS Centres, *Proc. European Computer Measurement Assoc. - 12*, Munich, Oct. 1984, 329-338. (METH)

[QSP82] A Modeling and Analysis Package (MAP), Quantitative System Performance, Seattle, WA., 1982. (TOOL)

[REI85] J. Reilly, "A Performance Engineering Case Study and Analysis," *Journal of Capacity Mgt.*, Vol. 2, No. 4, pp. 322-339. (EXPR)

[RID78] W.E. Riddle, J.C. Wileden, J.H. Sayler, A.R. Segal, A.M. Stavely, "Behavior Modeling During Software Design," *IEEE TSE* 4, 1978. (METH)

[RIV81] Elizabeth A. Rivet, "Performance Evaluation During System Selection," *Proc. Computer Measurement Group Conf XII*, Dec. 1981, 1-4. (EXPR)

[RZE86] W.E. Rzepka, P.C. Daley, "A Prototyping Tool to Assist in Requirements Engineering," *Proc. Hawaii Int. Conf. Sys. Sciences*, Jan. 1986, pp. 608-618. (TOOL)

[SAN77] J.W. Sanguinetti, "Performance Prediction in an Operating System Design Methodology," Ph.D. Dissertation, RSSM/32, University of Michigan, May 1977. (MDLG)

[SAN78] J.W. Sanguinetti, "A Formal Technique for Analyzing the Performance of Complex Systems," *Proc. Computer Performance Evaluation Users Group 14*, Boston, October 1978. (MDLG)

[SAN79] J.W. Sanguinetti, "A Technique for Integrating Simulation and System Design," *Proc. Conference on Simulation Measurement and Modeling of Computer Systems*, Boulder, August 1979. (MDLG)

[SAN84] J. Sanguinetti, "Program optimization for a pipelined machine: a case study," *Proc. SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, Cambridge, Aug. 1984, 88-95. (PGMS)

[SAU82a] C.H. Sauer, E.A. MacNair, J.F. Kurose, "The Research Queueing Package, Version 2: Introduction and Examples," Report RA128, IBM T.J. Watson Research Center, 1982. (TOOL)

[SAU82b] Charles H. Sauer, Edward A. MacNair, James F. Kurose, "The Research Queueing package: Past, Present, and Future," *Proc. National Computer Conf.*, Houston, June 1982, 273-280. (TOOL)

[SCH84] Paul A. Scheffer, William E. Rzepka, "A Large System Evaluation of SREM," *Proc. Int. Conf. Soft. Eng.* 7, IEEE Catalog No. 84C+12011-5, Orlando, Mar. 1984, pp. 172-180. (EXPR)

[SCH82] Schwetman, Herb, "Implementing the Mean Value Algorithm for the Solution of Queueing Network Models," Purdue University Report CSD-TR-355, February 1982. (TOOL)

[SEE82] Deborah J. Seeliger, "Application of PAWS in the Sperry Univac Environment: Phase 1," *Proc. Computer Measurement Group XIII*, San Diego, Dec. 1982, 200-219. (EXPR)

[SEV84] K.C. Sevcik, E. Lazowska, "Generating Queueing Network Models from High-level System Specifications," *Proc. Computer Measurement Group Conf. XV*, San Francisco, Dec. 1984,

103-197. (TOOL)

[SHA79] M. Shaw, "A Formal System for Specifying and Verifying Program Performance," Report CMU-CS-79-129, Carnegie-Mellon University, June 1979. (METH)

[SHO75] H.A. Sholl, T.L. Booth, "Software Performance Modeling Using Computation Structures," *IEEE Software Engineering* 1,4 December 1975. (MDLG)

[SHO86] H. Sholl, S. Kim, "An Approach to Performance Modelling as an Aid in Structuring Real-Time, Distributed System Software, *Proc. Hawaii Int. Conf. Sys. Sciences*, Jan. 1986, pp. 5-16. (MDLG)

[SIL82] H. Fred Silver, "A Pragmatic Approach to the Management of Software Development Life Cycles," *Proc. Computer Measurement Group Conf.*, San Diego, Dec. 1982, 142-155. (METH)

[SMI79a] C.U. Smith, J.C. Browne, "Performance Specifications and Analysis of Software Designs," *Proc. Conference on Simulation Measurement and Modeling of Computer Systems*, Boulder, August 1979. (MDLG)

[SMI79b] C.U. Smith, J.C. Browne, "Modeling Software Systems for Performance Predictions," *Proc. Computer Measurement Group X*, Dallas, December 1979. (MDLG)

[SMI80a] Connie U. Smith, J.C. Browne, "Aspects of Software Design Analysis: Concurrency and Blocking," *Proc. Performance 80* also in *ACM Performance Evaluation Review*, 9,2, Summer 1980. (MDLG)

[SMI80b] Connie U. Smith, "The Prediction and Evaluation of the Performance of Software from Extended Design Specifications," Ph.D. Dissertation, University of Texas at Austin, University Microfilms Pub. No. KRA81-00963, August 1980. (MDLG)

[SMI80c] Connie Smith, "Consider the Performance of Large Software Systems Before Implementation," *Proc. Computer Measurement Group Conference XI*, Dec., 1980, 138-144. (METH)

[SMI81] Connie U. Smith, "Software Performance Engineering," *Proc. Computer Measurement Group Conference XII*, Dec., 1981, 5-14. (METH)

[SMI82a] Connie U. Smith, "A Methodology for Predicting the Memory Management Overhead of New Software Systems," *Hawaii International Conf. on System Sciences*, 15, Jan. 1982, 200-209. (MDLG)

[SMI82b] Connie U. Smith, J.C. Browne, "Performance Engineering of Software Systems: A Case Study," *Proc. National Computer Conference*, Vol. 15, Houston, June 1982, 217-224. (EXPR)

[SMI82c] Connie U. Smith, David D. Loendorf, "Performance Analysis of Software for an MIMD Computer," *Proc. 1982 Conf. on Measurement and Modeling of Computer Systems*, Aug. 1982, 151-162. (MDLG)

[SMI84a] Connie U. Smith, "Effective Implementation of Software Performance Engineering," *Proc. European Computer Measurement Association* 12, Munich, Oct. 1984, 241-245, also in *Journal of Capacity Management*, Vol. 3, No. 1, 1985. (METH)

[SMI85a] Connie U. Smith, "Robust Models for the Performance Evaluation of Software/Hardware Designs," *Proc. Int. Conf. Timed Petri Nets*, Torino, July 1985, pp. 172-180. (MDLG)

- [SMI85b] Connie U. Smith (ed.), "Special Issue: Software Performance Engineering," *CMG Transactions*, Fall 1985.
- [SMI85c] Connie U. Smith, "Experience with Tools for Software Performance Engineering," *Proc. CMG 85*, Dallas, Dec. 1985, pp.411-417. (TOOL)
- [SMI86a] Connie U. Smith, "Synthesis Principles for High Performance Software," *Proc. Hawaii Int. Conf. on System Sciences*, Jan. 1986, pp. 17-27. (PRIN)
- [SMI86b] Connie U. Smith, "Independent General Principles for Constructing Responsive Software Systems," *ACM Trans. on Computer Systems*, 4,1, Feb. 1986, 1-31. (PRIN)
- [SMI86c] Connie U. Smith, "Evolution of Software Performance Engineering: A Survey," *Proc. FJCC*, Dallas, Nov. 1986, pp. 778-783. (METH)
- [SMI87b] Connie U. Smith, "General Principles for Performance Oriented Design," submitted to *CMG87*. (PRIN)
- [SMI87c] Connie U. Smith, "Better Performance through Engineering," *ComputerWorld*, Oct. 19, 1987, p. S15.
- [SMI87d] Connie U. Smith, "Software Fire Prevention," *Software News*, Nov. 1987, pp. 69-82.
- [SMI87e] Connie U. Smith (Chair), "Successful Software Performance Engineering," *Proc. CMG87*, Orlando, Dec. 1987.
- [SMI88] Connie U. Smith, "Applying Synthesis Principles to Create Responsive Software Systems," *IEEE TSE*, Oct. 88, (PRIN)
- [SQI86] D.J. Squillance, W.F. Hartman, "Applications Tuning in the IBM MVS Environment: Techniques, Tools, and Politics," *Proc. CMG 86*, Las Vegas, Dec. 1986, pp. 716-718. (PGMS)
- [TER84] Ralph Terkowitz, "Performance Management During Application Development," *Proc. Computer Measurement Group Conf. XV*, San Francisco, Dec. 1984, 467-470. (EXPR)
- [THO85] Thomasian, A., Bay, P. "Performance analysis of task systems using a queueing network model," *Proc. Int. Conf. Timed Petri Nets*, Torino, July 1985, 234-242. (MDLG)
- [VAN78] Van Tassel, D. *Program Style, Design, Efficiency, Debugging and Testing*, Prentice-Hall, Englewood Cliffs, NJ 1978. (PGMS)
- [VIA81] Duc J. Vianney, "Predicting Software Efficiency During the Design Process Using Software and Performance Metrics," *Proc. Computer Measurement Group Conf. XII*, Dec. 1981, 15-19. (MDLG)
- [VOL82] Elaine Volansky, "Performance Measurement of the Tracking and Data Relay Satellite," *Proc. Computer Measurement Group Conf. XIII*, San Diego, Dec. 1982, 121-129. (EXPR)
- [WAL79] Barbara Walter, "Programmer Productivity in Cobol Application Tuning," *Proc. Computer Measurement Group Conf. X*, Dallas, Dec. 1979, 477-516. (PGMS)
- [WET83] T.T. Wetmore, "Performance Analysis and Prediction from Computer Source Program Code," Ph.D. Dissertation, University of Connecticut, TR#CS-83-6, 1983. (MDLG)

[WON83] Wong, Kin L., "Design - A Tool for Software Performance Engineering" *CMG Transactions*, 41, September 1983. (TOOL)

[YAU81] S.S. Yau, C.C. Yang, S.M. Shatz, "An Approach to Distributed System Software Design," *IEEE TSE*, SE 7,4 July 1981. (MDLG)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1 AGENCY USE ONLY (Leave blank)		2 REPORT DATE June 1990		3 REPORT TYPE AND DATES COVERED Final	
4 TITLE AND SUBTITLE SOFTWARE PERFORMANCE ENGINEERING FOR REALTIME SYSTEMS				5 FUNDING NUMBERS C: N00039-86-C-0247	
6 AUTHOR(S)					
7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) L&S Computer Technology, Inc. P.O. Box 9802, Dept. 120 Austin, TX 78766				8 PERFORMING ORGANIZATION REPORT NUMBER	
9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Technology Arlington, VA 22217				10 SPONSORING/MONITORING AGENCY REPORT NUMBER NOSC TD 1830	
11 SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE	
13 ABSTRACT (Maximum 200 words) This document provides the materials that were used to teach a course in software performance engineering for realtime systems.					
14 SUBJECT TERMS software performance engineering (SPE)				15 NUMBER OF PAGES 305	
				16 PRICE CODE	
17 SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18 SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19 SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20 LIMITATION OF ABSTRACT SAME AS REPORT		